An Approach to a Unified Theory of Automata

By J. E. HOPCROFT* AND J. D. ULLMAN

(Manuscript received April 7, 1967)

A model of an automaton, called a balloan automaton is proposed, It consists of a finite control, which may bc deterministic or nondeterministic, an input tape which may be one way or two way, and an abstract, infinite memory, called the balloon, which can enter any of a countable nunber of states. There is assumed to be a recursive function which manipulates the state of the balloon, and another which passes a finite amount of information from the balloon to the finitc control.

A subset of the balloon automata is considered a closed class if it obeys two very simple closure properties. Certain closed elasses rccognize exactly the languages rccognized by such familiar automata as the pushdown automaton or staek automaton. Unfortunately, no closed class reeognizes the sets accepted by linear bounded automata or the time and tape complexity classes of Turing machines.

It is shown that many of the usual closure properties of languages accepted by the pushdown automaton, stack automaton, etc., hold for an arbitrary eloscd class of balloon automata. For example, the languages accepted by a elosed class of one-way, nondeterministic balloon automata are elosed under eoncatenation. Of special interest is the fact that a elosed class of two-way deterministie balloon automata is elosed under inverse g.s.m. mappings. This fact is not obvious, and was not known for all of the types of automata whieh form closed classes of balloon automata.

It should be emphasized that the purpose of this paper is not to propose another "model of a eomputer." Rather, we are proposing a method of proving the standard theorems about existing and future models. Hopefully, when a model is proposed in the future, one will simply show it equivalent to a closed class of balloon automata, and have many of the elosure properties automatically proven.

---

* Currently at Cornell University, Ithaca, N. Y.

I. INTRODUCTION

In the past, and especially recently, people have been examining various species of automata, perhaps as models of the compiling and translating processes, or for the insights they lend to computation. A partial list includes the Turing machine,[1] pushdown automaton,[2, 3, 4] deterministic pushdown automaton,[5] counter machine,[6, 7] stack automation, in all its forms, two-way,[8] one-way,[9, 10, 11] nonerasing,[12] deterministic and nondeterministic, the nested stack automaton,[13] and the time[14, 15] and tape[16, 17, 18] bounded Turing machines. This list is not meant to be a complete survey of past writings, and more can be expected in the future.

Many of the properties of each of the automaton classes mentioned are the same. For example, one would expect the set of languages accepted by each class to be closed under intersection with a regular set. Our plan is to propose a model of an automaton abstracting the common features of most of the models mentioned. We will define a *class* of automata to be a subset of the set of all such automata if it satisfies certain simple and physically meaningful closure properties. Then, from these closure properties, we will derive many of the common closure theorems which have been proven for the specific types of automata mentioned, and which, presumably, would be proven for future types.

The basic model is shown in Fig. 1. It consists of a two-way *input tape*, with end markers, a *finite control*, and an infinite storage of unspecified structure, called the *balloon*.

We assume that the states of the balloon are represented by the positive integers. A move of the automaton is a three-stage process. First, a recursive function is used to get a finite amount of informa-

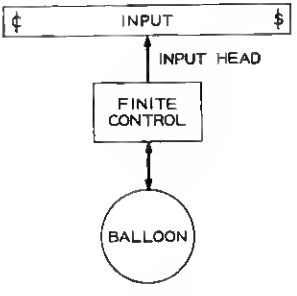

Fig. 1 — Balloon automaton.

tion from the balloon. Typically, this information is analogous to the symbol scanned by the storage head of an automaton with a tape memory. Second, based on the information from the balloon, the state of the finite control, and the symbol scanned by the input head, a new state of finite control and a direction of input head motion is determined. Third, based on the new state of finite control, and the current state of the balloon, a recursive function determines the next state of the balloon. Certain states of the finite control are *final states.* If the input causes the automaton to enter a final state, the input is *accepted.*

A subset of the set of balloon automata is called a *closed class,* or simply a *class,* if:

(*i*) It contains the finite automata.

(*ii*) If two automata are in the class, a third in the class can be found by associating in any way, the recursive functions getting information from the balloon and determining the next state of the balloon.

The latter condition is vague, but will be made formal.

Most, but not all, of the types of automata mentioned can be interpreted as classes under our definition. It seems that a type of automaton is a class if its definition involves only the ways in which the infinite storage may be locally manipulated. Sets such as the time and tape complexity classes of Turing machines do not form classes. With special emphasis, the linear-bounded automata unfortuately do not form a class in our formulation. Note that single changes in the next state of finite control function for a Turing machine may cause it to use much more time or tape than did the original machine, so condition (*ii*) would not be satisfied. Some of the automata, all two-way deterministic, which do form classes are:

(*i*) Pushdown automaton.
(*ii*) Stack automaton.
(*iii*) Nonerasing stack automaton.
(*iv*) Nested Stack automaton.
(*v*) Single counter machine.
(*vi*) Finite automaton.
(*vii*) Turing machine.

Our model shall be modified to treat nondeterministic and one-way input devices later in the paper. We have chosen two-way determinis-

tic devices to treat first because, with one exception, the theorems involved are quite straightforward.

II. THE TWO-WAY DETERMINISTIC BALLOON AUTOMATON

A balloon automaton consists of:

(*i*) A finite, nonempty set of *states*, S.

(*ii*) A finite set of *input symbols*, I, which includes $\not\epsilon$ and \$, the *left* and *right end-markers* of the input, respectively.

(*iii*) A set of *balloon states*, which is always the positive integers, denoted by Z.

(*iv*) A finite, nonempty set of integers, M, known as the *balloon information*.

(*v*) A total recursive function, h, from Z to M, known as tbe *balloon information function*.

(*vi*) A function g, with finite domain, $S \times I \times M$ and finite range $S \times \{-1, 0, +1\}$. We will also allow φ, the null set, in the range of g. We call g the *finite control function*.

(*vii*) A partial recursive function, f, from $S \times Z$ to Z, known as the *balloon control function*.

(*viii*) A subset, F, of S, called the *final states*.

(*ix*) A state q_0 in S, the *start state*. To simplify matters later, we will here assume that the start state is not a final state. The balloon automaton is denoted $(S, I, M, f, g, h, q_0, F)$.

We denote a *configuration* of the automaton $A = (S, I, M, f, g, h, q_0, F)$ by (q, w, j, i), where:

(*i*) q is a state of the finite control, in S.

(*ii*) w is in I^*. More specifically, $w = \not\epsilon a_1 a_2 \cdots a_n\$$, $n \geqq 0$, where for $1 \leqq k \leqq n$, a_k is in $I - \{\not\epsilon, \$\}$. Thus, $\not\epsilon$ marks the left end and \$ the right end. We call n the *length* of w. Endmarkers do not contribute to the length.

(*iii*) j is an integer between 0 and $n + 1$, denoting the position of the *input head* of A.

(*iv*) i is a positive integer, the state of the balloon.

As previously mentioned, a *move* of A is a three-stage process. Let (q_1, w, j_1, i_1) be a configuration of A, and the j_1th symbol of w be a. Let w, exclusive of endmarkers, consist of n symbols. We call $\not\epsilon$ the 0th symbol, \$ the $n + 1$st, and number the non-endmarker symbols from 1 to n from the left. Suppose $h(i_1) = m$. Then, find $g(q_1, a, m)$.

If it is φ, no move is possible. Suppose $g(q_1, a, m) = (q_2, d)$, where q_2 is in S and $d = -1, 0$, or $+1$. Then, compute, if possible, $f(q_2, i_1)$. Let it be i_2. If $j_2 = j_1 + d$ lies in the range 0 to $n + 1$, we say that a move is possible, and the next configuration is (q_2, w, j_2, i_2).

Note that $f(q_2, i_1)$ does not necessarily have a value. In that case, there is no move possible.

Intuitively, to make a move of A, we get what information we can from the balloon by calculating $h(i_1)$. Then, using g, we find the new state of finite control and direction of motion of the input head. Finally, using f, with the new state of finite control, we find the new balloon state.

If, from configuration (q_1, w, j_1, i_1), the next configuration of A is (q_2, w, j_2, i_2), we say: $(q_1, w, j_1, i_1) \vdash_A (q_2, w, j_2, i_2)$. If A can go from configuration (q_1, w, j_1, i_1) to configuration (q_2, w, j_2, i_2) by some number of moves, including zero moves, we say: $(q_1, w, j_1, i_1) \vdash_A^* (q_2, w, j_2, i_2)$.

Notation: We will, for a balloon control function f and state q in S, often use $f_q(i)$ for $f(q, i)$. Also define $\alpha^{(0)}$ to be the function from Z to Z such that $\alpha^{(0)}(i) = i$ for all i. Let $\alpha^{(j)}$, for integer $j \geqq 1$, be the function that takes i to j for all i in Z.

If $A = (S, I, M, f, g, h, q_0, F)$ is a balloon automaton, let the *tapes accepted by* A, denoted $T(A)$, be the set of w such that

$$(q_0, w, 0, 1) \vdash_A^* (q, w, j, i)$$

for some q in F, input head position, j, and balloon state, i. That is, starting in the start state with the input head at the left endmarker and the balloon in state 1, w must cause A to enter an accepting state.

Note that if g determines, in some configuration, that A enters state p, and p is an accepting state, but for the state of the balloon, i, $f_p(i)$ is not defined, then A has no next move, hence does not accept.

Let C be a subset of the set of all balloon automata. We say C is a *closed class*, hereafter shortened to *class*, if it satisfies the following two conditions:

I. $(S, I, M, f, g, h, q_0, F)$ is in C for any finite sets, $S, I, F \subseteq S$, q_0 in S, and arbitrary mapping g from $S \times I \times M$ to $(S \times \{-1, 0, +1\}) \cup \{\varphi\}$. We restrict h to be $\alpha^{(j)}$ for some $j \geqq 1$ and $M = \{j\}$. Also, for each q in S, f_q is $\alpha^{(k)}$ for some $k \geqq 0$.

II. Let $(S_1, I_1, M_1, f_1, g_1, h_1, q_1, F_1)$ and $(S_2, I_2, M_2, f_2, g_2, h_2, q_2, F_2)$ be in C. Then $(S_3, I_3, M_3, f_3, g_3, h_3, q_3, F_3)$ is in C if;

(*i*) S_3 and I_3 are arbitrary finite sets.
(*ii*) M_3 is the range of h_3.
(*iii*) q_3 is in S_3.
(*iv*) $F_3 \subseteq S_3$.
(*v*) g_3 is an arbitrary mapping from

$$S_3 \times I_3 \times M_3 \quad \text{to} \quad (S_3 \times \{-1, 0, +1\}) \cup \{\varphi\}.$$

(*vi*) For each q in S_3, $(f_3)_q$ is $(f_1)_p$ or $(f_2)_p$ for some p in S_1 or S_2, respectively.†
(*vii*) h_3 is a total recursive function such that if $h_3(i_1) \neq h_3(i_2)$ then either $h_1(i_1) \neq h_1(i_2)$ or $h_2(i_1) \neq h_2(i_2)$.

Intuitively, assumption (*i*) causes each of the regular sets to be accepted by some automaton in the class. Note that the function h is such that no information can be obtained from the balloon.

Assumption II insures that balloon control functions can be used interchangeably. The function associated with some state may be associated with none, one, or many states of a new automaton.

The information obtainable from h_3 is no more than the information obtainable from the combination of h_1 and h_2.

If C is a class of automata, then the set of languages which can be recognized by some automaton in C is called a *closed class of languages*, or simply a *class of languages*.

It should be clear that to every class, C, there corresponds a set of allowable balloon information functions, H_C. That is, a function, h, is in H_C if and only if it is the balloon information function for some automaton, A, in C. Likewise, there is a set of functions, F_C, which is the set of allowable balloon control functions restricted to a single state. That is, f is in F_C if and only if for some automaton A, in C, with balloon control function f_1, $f(i) = f_1(q, i)$ for some fixed state q of A.

Note that $\alpha^{(i)}$ is in H_C for all $i \geqq 1$, and $\alpha^{(i)}$ for $i \geqq 0$ is in F_C, or any class C. We can use the following obvious result:

Lemma 1: Let h be in H_C and $f_1, f_2, \cdots, f_s$ be in F_C. Let $S = \{q_1, q_2, \cdots, q_s\}$, I be an arbitrary set of inputs including ¢ and \$, M the range of h, g an arbitrary map from $S \times I \times M$ to $(S \times \{-1, 0, +1\}) \cup \{\varphi\}$, and $F \subseteq S$. Then $(S, I, M, f, g, h, q_k, F)$ is in C for any q_k in S, and f defined by $f(q_j, i) = f_j(i)$ for all i.

Proof: Let $B_0 = (S_0, I_0, M, d_0, g_0, h, p_0, F_0)$ be an automaton in C

† Recall $(f_3)_q$ is by definition the function such that $(f_3)_q(i) = f_3(q,i)$ for all i. Likewise $(f_1)_p$ and $(f_2)_p$.

with balloon information function h, and $A_i = (S_i, I_i, M_i, d_i, g_i, h_i, p_i, F_i)$ be automata in C such that for each i, $1 \leqq i \leqq s$, there is a state, r_i, in S_i, such that $d_i(r_i, j) = f_i(j)$ for all j.

For $1 \leqq i \leqq s$, define B_i from B_{i-1} and A_i according to rule II. Let $B_i = (S, I, M, e_i, g, h, q_k, F)$, where $(e_i)_{a_j} = f_j$ if $j \leqq i$, and $(e_i)_{a_j} = f_i$ if $j > i$. Surely, $e_s = f$, so B_s is our desired balloon automaton.

Lemma 2: Let $A = (S, I, M, f, g, h, q_0, F)$ be an automaton in Class C. Let $A_1 = (S_1, I_1, M, f_1, g_1, h, q_1, F_1)$ be such that for every p in S_1, $(f_1)_p$ is either $\alpha^{(i)}$ for some $i \geqq 0$ or f_q for some q in S. Then A_1 is in class C.

Proof: All f_q, for q in S are in F_C, and h is in H_C. Also, $\alpha^{(i)}$ is in F_C for all $i \geqq 0$ by rule (I). A_1 is in class C by Lemma 1.

We should comment that it is quite natural to force $\alpha^{(0)}$ to be in F_C for any class, C. Intuitively, the consequence is that an automaton may do computation in its finite control without affecting the infinite portion of storage. We also force $\alpha^{(i)}$, for $i \geqq 1$ to be in F_C. These mappings enable us to reset the infinite memory to any given state. Their use will be apparent, but their justification is not so clear. We only observe that for any of the seven types of automata mentioned, suitable modifications, which do not change the power of the devices, can be made, so that a device can reset itself to a given state.

For example, a Turing machine can surely erase its tape and print any given tape string thereon. Of course, it takes more than one move to do so, but this fact should not concern us. Even a nonerasing stack automaton can print a dummy "end of stack" marker at the top of stack to simulate an erasure of the stack.

Example: Let us indicate how to interpret a two way deterministic pushdown automaton as a class of balloon automata. We will not give a formal definition here. Most readers should be familiar with the concept of an automaton with pushdown storage, usually taken to be nondeterministic, with a one-way input. The two-way, deterministic variety is defined formally in Ref. 4.

Informally, the infinite storage is a pushdown tape, of which the automaton can at any time read only the top symbol. The pushdown tape can be altered by erasing the top symbol, or by adding a symbol to the top of the list.† The pushdown automaton has a finite control, input tape and input head, similar to these portions of a balloon automaton.

† The model of Ref. 4 allows one to add any finite number of symbols, but this mode is equivalent to adding one at a time.

We shall not formally prove that there is a closed class of balloon automata accepting exactly the sets accepted by two-way, deterministic pushdown automata. We shall merely give the sets H_C and F_C of balloon information and balloon control functions, and indicate how they reflect the pushdown structure of storage. We shall also indicate how any balloon automaton in the class can be simulated by a two-way, deterministic pushdown automaton.

To begin, we shall assign the usual Gödel numbering to pushdown tapes. That is, let the allowable pushdown symbols be $Z_1, Z_2, \cdots, Z_m$. Represent the pushdown list $Z_{i_1}Z_{i_2} \cdots Z_{i_k}$ by $2^{i_1}3^{i_2}5^{i_3} \cdots [\pi(k)]^{i_k}$. Here $\pi(i)$ stands for the ith prime. ($\pi(1) = 2$, $\pi(2) = 3$, $\pi(3) = 5$, etc.). Define $\mu(i)$, for $i \neq 1$, to be the number of the largest prime dividing i. and define $\kappa(i)$ to be the number of times $\pi(\mu(i))$ divides i. Let $\mu(1) = 0$: $\kappa(1)$ also is 0. For example, $\mu(75) = 3$, because the third prime, 5, is the largest prime dividing 75. $\kappa(75) = 2$, since 5 divides 75 twice.

Define F to be a set of recursive functions given by:

(*i*) $\alpha^{(i)}$, for all $i \geqq 0$ is in F.

(*ii*) For any integer, d, the function f, defined by $f(i) = i[\pi(\mu(i) + 1)]^d$ $i \geqq 1$, is in F. Note that $f(i)$ finds the prime above the largest prime dividing i, and multiplies i by that prime, raised to the power d.

(*iii*) The function f, given by $f(1)$ is undefined, $f(i) = i/[\pi(\mu(i))]^{\kappa(i)}$, $i > 1$, is in F. This function divides i by the largest prime dividing i, as many times as it divides i.

The set H_C includes $\alpha^{(i)}$ for $i \geqq 1$. H_C also includes any total recursive function h if there is an integer d such that $h(i) \neq h(j)$ only if $\kappa(i) \neq \kappa(j)$, and at least one of $\kappa(i)$ and $\kappa(j)$ is equal to or less than d.

Let a given pushdown automaton, P, have m pushdown symbols, $Z_1, Z_2, \cdots, Z_m$. We will find a balloon automaton, A, whose balloon information function is in H, and whose balloon control function for any given state is found in F. The balloon information function, h, will have $h(i) \neq h(j)$ if $\kappa(i) \neq \kappa(j)$ for $\kappa(i)$ and $\kappa(j)$ each $\leqq m$. According to the Gödel numbering of pushdown tapes we mentioned, $h(i)$ will always indicate the top pushdown symbol of the tape numbered i, provided tape i involves symbols $Z_1, Z_2, \cdots, Z_m$ only.

Based on the top pushdown symbol, the state of P's finite control (which is carried in the finite control of A), and the symbol scanned by A's input head, A can move its input head, and change state according to what P would do. A may then have to adjust its balloon state to simulate a change in P's pushdown store. If P does nothing to the pushdown store, the function $\alpha^{(0)}$ serves. If P erases the top symbol,

the function f, in F by rule (*iii*) must be used. If P prints Z_j on top of the pushdown list, the function f_j in F by rule (*ii*), with $d = j$ suffices.

There is a subset, C, of balloon automata defined by placing an automaton in C exactly if its balloon information function is in H and its balloon control function, restricted to any particular state, is in F. We claim that C is a closed class. Surely every balloon automaton defined by rule I of the definition is in C.

In rule II, we have two automata, A_1 and A_2, in C, and must show that a third automaton, A_3, constructed from A_1 and A_2 is also in C. Certainly, the balloon control functions of A_3 are in F. Let h_1 and h_2 be the balloon information functions of A_1 and A_2, respectively. Assume h_1 and h_2 are in H. Let h_3 be the information function of A_3. Suppose $h_3(i) \neq h_3(j)$. Then either $h_1(i) \neq h_1(j)$ or $h_2(i) \neq h_2(j)$, by rule II. In either case, $\kappa(i) \neq \kappa(j)$. Also, since h_1 and h_2 are in H, we can find an integer, d, such that one of $\kappa(i)$ and $\kappa(j)$ is $\leqq d$. Thus, h_3 is in H.

Now we must show that any balloon automaton in C can be simulated by a two-way pushdown automaton. The details of simulating the finite control and input head of the balloon automaton can be left to the reader. We shall only discuss how the balloon can be simulated.

Let $A = (S, I, M, f, g, h, q_0, F)$ be in class C. Some f_q, for q in S, may multiply the ballon state, i, by a prime raised to some power, d. Note that this prime cannot divide i. Let d_1 be the maximum such d. Some f_q may be $\alpha^{(j)}$ for $j \geqq 1$. Now, let d be the maximum number of times a prime divides j, and let d_2 be the maximum such d. Finally, let d_3 be max (d_1, d_2).

The pushdown automaton, P, simulating A, will have $d_3 + 2$ pushdown symbols, $X, Z_0, Z_1, \cdots, Z_{d_3}$. X will mark the bottom of the pushdown list. For some k, each integer, i, can be expressed in prime factors as $[\pi(1)]^{i_1}[\pi(2)]^{i_2} \cdots [\pi(k)]^{i_k}$, where each i_j, $1 \leqq j \leqq k$, lies between 0 and d_3, but $i_k \neq 0$. Then i will be represented by pushdown tape $XZ_{i_1}Z_{i_2} \cdots Z_{i_k}$. It should be clear that if $h(i) \neq h(j)$, then the tapes representing i and j have different top (rightmost) symbols.

Suppose A uses a balloon control function that is in F according to rule (*iii*). Then P erases the top pushdown symbol. P must also erase from the top, any occurrences of Z_0. Suppose A uses a balloon control function that is in F by rule (*ii*), with some particular value of d. Surely $1 \leqq d \leqq d_3$. P must print Z_d on the top of its pushdown list. Finally, if A uses balloon control function $\alpha^{(i)}$, $i \geqq 1$, P erases its tape down to X, then prints $Z_{i_1}Z_{i_2} \cdots Z_{i_k}$ on its stack, where $i = [\pi(1)]^{i_1}[\pi(2)]^{i_2} \cdots [\pi(k)]^{i_k}$. Note that by definition of d_3, we must have $i_j \leqq d_3$ for all j.

From the way F is defined, it is easy to show that for any automaton with balloon control functions chosen from F, there is some d_3, chosen as above, such that if the balloon can enter a state i, then no prime divides i more than d_3 times. Thus P, as above, with $d_3 + 2$ pushdown symbols, can simulate the balloon of A.

III. SOME THEOREMS ABOUT TWO-WAY DETERMINISTIC BALLOON AUTOMATA

We have spent time defining closed classes of automata. Our goal is not so much to talk about the classes themselves, but rather about the properties of the closed classes of languages that they define. Let us begin with a not unexpected result.

Theorem 1: Let $A = (S, I, M, f, g, h, q_0, F)$ *be a balloon automaton. Then* $L = T(A)$ *is a recursively enumerable set.*

Proof: We shall describe, informally, a Turing machine recognizing L. First, we have assumed f to be partial recursive and h total recursive. Hence, there is a Turing machine, T_0 which, given a block of i 1's on its single tape will halt with $h(i)$ 1's on its tape. Likewise, let $S = \{q_1, q_2, \cdots, q_s\}$. Then there are Turing machines $T_1, T_2, \cdots, T_s$ such that given i 1's on its tape, T_j will eventually halt with $f_{q_j}(i)$ 1's on its tape if $f_{q_j}(i)$ is defined, and not halt otherwise, for each j, $1 \leqq j \leqq s$.

We will now construct a Turing machine, T, recognizing L, by simulating A. T is shown in Fig. 2. It has a read only input tape with endmarkers, and two storage tapes. The first is used to store the state of the balloon of A.

The second is used for the computation of h and f. The finite control of T will store the state of A's finite control.

Initially, the input head of T is at the left endmarker. Its finite control records that A's finite control is in state q_0. Storage tape 1

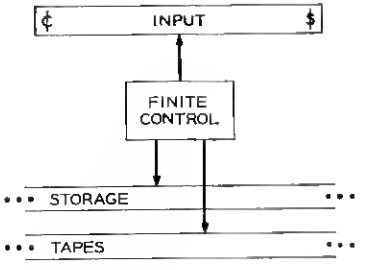

Fig. 2 — Turing machine T.

has a single 1 on it, corresponding to the initial state of A's balloon, and tape 2 is blank.

Suppose T has simulated some number of A's moves with given input. That is, T's input head is at the same position as A's would be after that number of moves. The finite control of T holds the state of A's finite control, and tape 1 holds the state of A's balloon. We will show how T simulates the next move of A, if A has a next move.

(*i*) Copy tape 1 onto tape 2.

(*ii*) Simulate T_0 on tape 2. When T_0 halts, suppose there are m 1's on tape 2 at that time.

(*iii*) Suppose T has recorded that q is the state of A's finite control. The symbol scanned by T's input head is a. Then T moves according to $g(q, a, m)$. If $g(q, a, m) = \phi$, T never completes simulation of the move of A. If $g(q, a, m) = (p, d)$, T records p as the state of A's finite control replacing q. T moves its input head in the direction indicated by d. If to do so would cause the input head to leave the input, T makes no move, but halts without accepting.

(*iv*) If T has simulated the first two stages of A's move, it again copies tape 1 onto tape 2. Let p be q_j for some j, $1 \leqq j \leqq s$. Then T simulates T_j on tape 2. If f_{q_i} is defined for the number of 1's on tape 2, T_j will eventually print on tape 2 a number of 1's equal to the new state. If not, T will not halt, hence no move of A is simulated.

(*v*) Finally, T copies tape 2 onto tape 1 and prepares to simulate another move of A. However, if the three phases of the move of A have each been successfully simulated, and p is in F, then T simulates no further moves of A, but rather, halts and accepts.

It is straightforward to see that T will simulate all moves of A, and will accept exactly when A reaches an accepting configuration.

We shall now consider three properties of closed classes of languages. These properties are that closed classes of languages are closed under reversal, intersection and inverse g.s.m. mappings. The third property is perhaps the only one in the paper that is difficult to prove.

Theorem 2: Let C be a class of automata. Let $L = T(A)$ for some $A = (S, I, M, f, g, h, q_1, F)$ in C. For any $w = \not{c}a_1a_2 \cdots a_n\$$ in I^*, define $w^r = \not{c}a_na_{n-1} \cdots a_1\,\$$. Define $L^r = \{w \mid w^r \text{ is in } L\}$. Then there is an automaton, A_1, in C such that $L^r = T(A_1)$.

Proof: Let $S = \{q_1, q_2, \cdots q_s\}$. Define $S_1 = \{q_1, q_2, \cdots q_{s+1}\}$, and $A_1 = (S_1, I, M, f_1, g_1, h, q_{s+1}, F)$. We define f_1 and g_1 as follows:

(*i*) $(f_1)_q = f_q$ for q in S.
(*ii*) $(f_1)_{q_{s+1}} = \alpha^{(0)}$.
(*iii*) Suppose $g(q, a, m) = (p, d)$ for some q in S, m in M, and a in $I - \{\not\$, \$\}$. Then $g_1(q, a, m) = (p, \bar{d})$ where $\bar{d} = +1, 0$ or -1 as $d = -1, 0$ or $+1$, respectively.
(*iv*) Suppose $g(q, \not\$, m) = (p, d)$, for q in S and m in M. Then $g_1(q, \$, m) = (p, \bar{d})$. If $g(q, \$, m) = (p, d)$, then $g_1(q, \not\$, m) = (p, \bar{d})$.
(*v*) $g_1(q_{s+1}, a, m) = (q_{s+1}, +1)$ for m in M and a in $I - \{\$\}$.
(*vi*) $g_1(q_{s+1}, \$, m) = (p, \bar{d})$ for m in M, where $g(q_1, \not\$, m) = (p, d)$.
(*vii*) g_1 is ϕ if not defined by (*iii*)–(*vi*).

A_1 is in class C by Lemma 2. We must show that $T(A_1) = L^r$. Let the input to A_1 be w, of length n. By rules (*ii*) and (*v*) it is seen tbat $(q_{s+1}, w, 0, 1) \overset{*}{\underset{A_1}{\vdash}} (q_{s+1}, w, n + 1, 1)$. From that configuration, A_1 never returns to state q_{s+1}, but simulates A with the direction of input head reversed.

That is, by rules (*i*) and (*vi*), $(q_{s+1}, w, n + 1, 1) \underset{A_1}{\vdash} (p, w, j, i)$ if and only if $(q_1, w^r, 0, 1) \underset{A}{\vdash} (p, w^r, n + 1 - j, i)$. Also, by rules (*i*), (*iii*) and (*iv*), for any q and p in S, integers i_1, i_2, j_1, j_2, with j_1 and j_2 between 0 and $n + 1$, $(q, w, j_1, i_1) \underset{A_1}{\vdash} (p, w, j_2, i_2)$ if and only if $(q, w^r, n + 1 - j_1, i_1) \underset{A}{\vdash} (p, w^r, n + 1 - j_2, i_2)$. Thus, by induction on the number of moves made by A, starting with one move,

$$(q_{s+1}, w, 0, 1) \overset{*}{\underset{A_1}{\vdash}} (p, w, j, i)$$

if and only if $(q_1, w^r, 0, 1) \overset{*}{\underset{A}{\vdash}} (p, w^r, n + 1 - j, i)$. We conclude that A_1 accepts its input, w, if and only if A accepts w^r. That is, $T(A_1) = L^r$. Note that A could not accept without making a move, since q_1 is not an accepting state.

Notation: Let h_1 and h_2 be balloon information functions, with ranges M_1 and M_2, respectively. Let M_1 have maximum element k. Define $h_1 \cdot h_2$ to be the function $[h_1 \cdot h_2](i) = h_1(i) + (k + 1)h_2(i)$. Define $M_1 \cdot M_2$ to be the range of $h_1 \cdot h_2$. We will also need the functions which are partial inverses of the $\cdot$ operator. So, we define $\sigma_1(k, j) = j$ modulo $k + 1$, and $\sigma_2(k, j) = [j/(k + 1)]$.† If k is as above, and $j = [h_1 \cdot h_2](i)$, then $\sigma_1(k, j) = h_1(i)$ and $\sigma_2(k, j) = h_2(i)$.

Note that according to the definition of closed class, if h_1 and h_2 are in H_C, then $h_1 \cdot h_2$ is in H_C for any closed class, C.

Theorem 3: If $A_1 = (S_1, I_1, M_1, f_1, g_1, h_1, q_1, F_1)$ *and* $A_2 = (S_2, I_2, M_2, f_2, g_2, h_2, q_2, F_2)$ *are automata in class* C, *then there is an automaton* A_3 *in* C *accepting* $L = T(A_1) \cap T(A_2)$.

† $[x]$ is the integer part of x.

Proof: By a simple application of Lemma 2, we can find automata accepting $T(A_1)$ and $T(A_2)$, each of whose set of input symbols is $I_1 \cup I_2$. So, we will assume that $I_1 = I_2 = I$. Likewise, from Lemma 2, we can assume S_1 and S_2 are disjoint. We construct a third automaton, $A_3 = (S_3, I, M_3, f_3, g_3, h_3, q_1, F_2)$. Here, $S_3 = S_1 \cup S_2 \cup \{q_3\}$, where q_3 is not in S_1 or S_2. Also, $M_3 = M_1 \cdot M_2$ and $h_3 = h_1 \cdot h_2$. We define f_3 and g_3 as follows:

(*i*) If q is in S_1, then $(f_3)_q = (f_1)_q$. If q is in S_2, then $(f_3)_q = (f_2)_q$.
(*ii*) $(f_3)_{q_3} = \alpha^{(1)}$.
(*iii*) Let k be the largest element in M_1, and let m be in M_3, with $m_1 = \sigma_1(k, m)$ and $m_2 = \sigma_2(k, m)$. Let a be in I. Suppose q is in S_1 but not in F_1, and $g_1(q, a, m_1) = (p, d)$. Then $g_3(q, a, m) = (p, d)$. If q is in F_1, $g_3(q, a, m) = (q_3, 0)$.

Suppose q is in S_2, instead, and $g_2(q, a, m_2) = (p, d)$. Then $g_3(q, a, m) = (p, d)$.

(*iv*) $g_3(q_3, a, m) = (q_3, -1)$, for all a in $I - \{\phi\}$ and m in M_3.
(*v*) $g_3(q_3, \phi, m) = (p, d)$ if $g_2(q_2, \phi, m_2) = (p, d)$, where m_2 is as in (*iii*).

From rules (*i*) and (*iii*) it is clear that until A_1 enters an accepting state, A_3 enters a configuration (q, w, j, i), q in S_1, if and only if A_1 would enter that configuration. If $(q_1, w, 0, 1) \overset{*}{\underset{A_1}{\vdash}} (p, w, j, i)$, where p is in F_1, and no accepting state has been previously entered, then by rules (*i*) and (*iii*), $(q_1, w, 0, 1) \overset{*}{\underset{A_3}{\vdash}} (p, w, j, i) \underset{A_3}{\vdash} (q_3, w, j, 1)$. If w is not accepted by A_1, then A_3 will never enter state q_3.

By rules (*ii*) and (*iv*) $(q_3, w, j, 1) \overset{*}{\underset{A_3}{\vdash}} (q_3, w, 0, 1)$. By rules (*i*) and (*v*), $(q_3, w, 0, 1) \underset{A_3}{\vdash} (q, w, j, i)$ if and only if $(q_2, w, 0, 1) \underset{A_2}{\vdash} (q, w, j, i)$. From this point, A_3 simulates A_2 in a straightforward manner, entering an accepting state with w as input if and only if A_2 does. Thus, in order for A_3 to accept w, both A_1 and A_2 must accept it, and whenever these accept w, A_3 will likewise accept w. In other words, $T(A_3) = T(A_1) \cap T(A_2)$.

By part II of the definition, and Lemma 2, A_3 is in class C.

We are now going to prove a theorem on inverse g.s.m. mappings. A *generalized sequential machine* (g.s.m.) is a finite state transducer.[19] It is usually defined as a 6-tuple, $G = (K, \Sigma, \Delta, \delta, \lambda, p_0)$. K, Σ and Δ are the finite sets of *states*, *input symbols* and *output symbols*, respectively. δ is a mapping from $K \times \Sigma$ to K, and λ is a mapping from $K \times \Sigma$ to Δ^*. Lastly, p_0 is in K and is called the *start state*. We extend δ and λ to domain $K \times \Sigma^*$ as follows: $\delta(q, \epsilon) = q$ and $\lambda(q, \epsilon) = \epsilon$, for all q in K. For w in Σ^* and a in Σ, $\delta(q, wa) = \delta(\delta(q, w), a)$ and $\lambda(q, wa) = \lambda(q, w)\lambda(\delta(q, w), a)$. Define $G(w) = \lambda(p_0, w)$.

We can define a function γ for the g.s.m. G, as above. γ maps $K \times \Sigma^*$ to the subsets of K. If q is in K and w is in Σ^*, then

$$\gamma(q, w) = \{p \mid \delta(p, w) = q\}.$$

For w in Σ^* and a in Σ, given $\gamma(q, w)$, we can find $\gamma(q, aw)$ by: $\gamma(q, aw) = \{p \mid \text{for some } p_1 \text{ in } \gamma(q, w),\ \delta(p, a) = p_1\}$.

We intend to prove that if A is a balloon automaton of class C, and G is a g.s.m., then there is an automaton, A_1, in C, such that $T(A_1) = \{\not\!\!\$w\$ \mid \text{if } G(w) = w_1, \text{ then } \not\!\!\$w_1\$ \text{ is in } T(A)\}$. We need an auxiliary definition and a lemma.

A *two-way finite automaton*[20] is a device with a two way, read only input tape and a finite control. Formally, the device is denoted $A = (K, \Sigma, \delta, p_0, F)$. K and Σ are finite sets of *states* and *input symbols*, respectively. Σ always includes $\not\!\!\$$ and \$, the left and right endmarkers of the input, respectively. $F \subseteq K$ is the set of *final states*, and p_0, in K, is the *start state*. δ maps $K \times \Sigma$ to $K \times \{-1, +1\}$. Intuitively, if $\delta(q, a) = (p, d)$, then A, scanning a on its input, in state q, goes to state p, and moves its input head left or right, depending on whether $d = -1$ or $+1$.

We denote a *configuration* of A, with input w, by (q, w, i). We assume w can be written as $\not\!\!\$w_1\$$, where w_1 is in $(\Sigma - \{\not\!\!\$, \$\})^*$. Let w_1 consist of n symbols. The position of the input head is indicated by i. That is, $i = 0$ if the input head is scanning $\not\!\!\$$, if $i = n + 1$, the head scans \$, and if $1 \leqq i \leqq n$, the head scans the ith symbol of w_1, counting from the left. Thus, $\not\!\!\$$ is the zeroth symbol of w, and \$ the $n + 1$st. Of course, q is the current state of A.

Say that $(q_1, w, i_1) \vdash_A (q_2, w, i_2)$ if a is the i_1th symbol of w, $\delta(q_1, a) = (q_2, d)$ and $i_2 = i_1 + d$. However, we must have $0 \leqq i_2 \leqq n + 1$. We define the relation $\vdash_A^*$ by $(q, w, i) \vdash_A^* (q, w, i)$, for any configuration, (q, w, i), of A, and $(q_1, w, i_1) \vdash_A^* (q_m, w, i_m)$ if there are configurations (q_2, w, i_2), (q_3, w, i_3), $\cdots$, (q_{m-1}, w, i_{m-1}) such that for $1 \leqq j < m$, $(q_j, w, i_j) \vdash_A (q_{j+1}, w, i_{j+1})$. Although we are not concerned with acceptance by two-way finite automata, (they accept the regular sets, as is well known) we will define the tapes accepted by A, denoted $T(A)$, to be $\{w \mid w \text{ in } \not\!\!\$(\Sigma - \{\not\!\!\$, \$\})^*\$, (p_0, w, 0) \vdash_A^* (p, w, i)$ for some p in F and integer, $i\}$.

Lemma 3: Let $G = (K, \Sigma, \Delta, \delta, \lambda, p_0)$ be a g.s.m., with $\not\!\!\$$ and \$ not in Σ. Then, we can construct a two-way finite automaton,

$$A = (K_1, \Sigma \cup \{\not\!\!\$, \$\}, \delta_1, q_0, F)$$

with the following properties:

(i) K_1 *is expressed as* $K_2 \times K$. *Elements of* K_1 *are denoted* $[q, p]$ *where* q *is in* K_2, p *in* K.

(ii) q_1 *and* q_2 *are particular elements of* K_2.

(iii) Let $w = a_1 a_2 \cdots a_n$ *be in* Σ^*, *each* a_k *in* Σ, $1 \leqq k \leqq n$. *Suppose* $\delta(p_0, a_1 a_2 \cdots a_{i-1}) = p$, $i \geqq 2$. *Then*

$$([q_1, p], ¢w\$, i) \overset{*}{\underset{A}{|}} ([q_2, q], ¢w\$, i - 1),$$

where $\delta(p_0, a_1 a_2 \cdots a_{i-2}) = q$. *Never is* q_1 *or* q_2 *the first component of state of* A, *except for the first and last configurations.*

(iv) q_0 *and* F *are irrelevant, since the lemma concerns, not the recognizing power, but the structure of two-way finite automata.*

Proof: This lemma was essentially proven in Ref. 11, with direction of input head reversed. We shall, therefore, not give a formal proof, but just sketch the argument. The result in Ref. 11 did not involve the function δ of a g.s.m., but another function which had the properties needed, properties which δ has. These properties are:

(i) $\delta(q, w)$ is unique for q in K, w in Σ^*.

(ii) If γ is defined as in the definition of the g.s.m., and p_1 and p_2 are in K, $p_1 \neq p_2$, then for any w in Σ^*, $w \neq \epsilon$, $\gamma(p_1, w)$ and $\gamma(p_2, w)$ are disjoint. (For if p were in both, then $\delta(p, w) = p_1 = p_2$, violating *(i)*.)

(iii) If p_3 is in $\gamma(p_1, w)$ and p_3 in $\gamma(p_2, w)$, and $w = w_1 w_2$ with $w_2 \neq \epsilon$, then $\delta(p_3, w_1) \neq \delta(p_4, w_1)$. (For if not, let $\delta(p_3, w_1) = \delta(p_4, w_1) = p$. Then $\gamma(p_1, w_2)$ and $\gamma(p_2, w_2)$ each contain p, and $w_2 \neq \epsilon$, violating *(ii)*.)

(iv) If p_1 and p_2 are in $\gamma(p, w)$, then $\delta(p_1, w) = \delta(p_2, w) = p$, by definition of γ.

We will now sketch the design of A. Let $¢w\$$ be its input, $w = a_1 a_2 \cdots a_n$, as in the statement of the lemma. Suppose the input head of A is scanning a_i, and A is in state $[q_1, p]$. Presumably,

$$\delta(p_0, a_1 a_2 \cdots a_{i-1}) = p.$$

A moves its input head left, and computes $\gamma(p, a_{i-1})$. If $\gamma(p, a_{i-1})$ contains a single element, p_1, then p_1 must be $\delta(p_0, a_1 a_2 \cdots a_{i-2})$. A can easily enter configuration $([q_2, p_1], w, i - 1)$.

It is not possible that $\gamma(p, a_{i-1})$ is empty. Suppose $\gamma(p, a_{i-1})$ contains r elements, $r > 1$. Let these be $p_1, p_2, \cdots, p_r$. A moves left. For $j = i - 2, i - 3, i - 4, \cdots$ it successively computes

$$\gamma(p_k, a_j a_{j+1} \cdots a_{i-2})$$

from $\gamma(p_k, a_{j+1} a_{j+2} \cdots a_{i-2})$ for $1 \leqq k \leqq r$. Unless the process terminates, in one of two ways we will describe, A then drops $\gamma(p_k, a_{j+1} a_{j+2}$

$\cdots a_{i-2}$) from memory. Given G, we can find an upper bound on r, so the amount of information stored in A's finite control is bounded.

(*i*) Suppose that for some largest j, for only one value of k, say $k = m$, is $\gamma(p_k, a_j a_{j+1} \cdots a_{i-2})$ nonempty. Then surely p_m is $\delta(p_0, a_1 a_2 \cdots a_{i-2})$. A must find its way back to position $i - 1$. Presumably, one can find k_1 and k_2 such that $\gamma(p_{k_1}, a_{j+1} a_{j+2} \cdots a_{i-2})$ and $\gamma(p_{k_2}, a_{j+1} a_{j+2} \cdots a_{i-2})$ are not empty. Choose s_1 and s_2 from these sets, respectively. A then moves right, computing $\delta(s_1, a_{j+1} a_{j+2} \cdots a_l)$ and $\delta(s_2, a_{j+1} a_{j+2} \cdots a_l)$ for $l = j + 1, j + 2, \cdots$. By comments (*iii*) and (*iv*) above, we will not have $\delta(s_1, a_{j+1} a_{j+2} \cdots a_l) = \delta(s_2, a_{j+1} a_{j+2} \cdots a_l)$ until $l = i - 1$. A is thus positioned properly, and can enter configuration

$$([q_2, p_m], w, i - 1).$$

(*ii*) Suppose that no j satisfies condition (*i*). Then A will eventually reach the left endmarker. It must be that for some m, p_0 is in $\gamma(p_m, a_1 a_2 \cdots a_{i-2})$. Thus, p_m is $\delta(p_0, a_1 a_2 \cdots a_{i-2})$. A must find its way back to position $i - 1$. So, A chooses s_1 and s_2 in $\gamma(p_{k_1}, a_1 a_2 \cdots a_{i-2})$ and $\gamma(p_{k_2}, a_1 a_2 \cdots a_{i-2})$ for some $k_1 \neq k_2$. A moves right, successively computing $\delta(s_1, a_1 a_2 \cdots a_l)$ and $\delta(s_2, a_1 a_2 \cdots a_l)$ for $l = 1, 2, \cdots$. When $\delta(s_1, a_1 a_2 \cdots a_l) = \delta(s_2, a_1 a_2 \cdots a_l)$, we must have $l = i - 1$. A easily enters configuration $[q_2, p_m], w, i - 1)$.

Theorem 4: Let $A_1 = (S_1, I_1, M, f_1, g_1, h, r_1, F_1)$ be a balloon automaton in class C. Let $G = (K, \Sigma, \Delta, \delta, \lambda, p_0)$ be a g.s.m., where $\Delta = I_1 - \{¢, \$\}$. Then there is an automaton, A_2 in class C, such that

$$T(A_2) = \{¢w\$ \mid ¢G(w)\$ \text{ is in } T(A_1).\}.$$

$T(A_2)$ is commonly called an inverse g.s.m. mapping of $T(A_1)$.

Proof: Let $A = (K_1, \Sigma \cup \{¢, \$\}, \delta_1, q_0, F)$ be the two-way finite automaton constructed from G in Lemma 3. Let $A_2 = (S_2, I_2, M, f_2, g_2, h, r_2, F_2)$, where $I_2 = \Sigma \cup \{¢, \$\}$. Let $S_2 = \{[q, p, r, u, l, k] \mid q$ in K_2, p in K, r in S_1, u a string in $(I_1 - \{¢, \$\})^*$, of length at most max $(|\lambda(s, a)|$ for s in K, a in $\Sigma)$, l an integer between 0 and $|u|$, and k an integer between 1 and 8$\}$.† K_2 is defined as in Lemma 3, as are its particular elements, q_1 and q_2. $r_2 = [q_2, p_0, r_1, \epsilon, 0, 1]$. F_2 is the set of all states in S_1 whose last component is 8.

We shall call the last component of states in S_2 the *pointer*. It indicates, among other things, if A_2 is simulating A, A_1 or G. The first component is part of a state of A. It is needed because A_2 may move its

† $|x|$ denotes the length of string x.

head left to simulate A_1. In that case, the routine A is needed to determine the state of G at the new position of A_2's input head. The second component of A_2's state indicates what state G would be in if it had processed whatever is to the left of A_2's input head. The third component is the state of A_1. The fourth component is the output when the input to G is the symbol currently scanned by A_2's input head. The fifth component indicates where, among the symbols of the fourth component, A_1's input head would be. In Fig. 3, the construction of A_2 is symbolically indicated.

We define f_2 by:

(*i*) $(f_2)_{[q,p,r,u,t,k]} = \alpha^{(0)}$ for $k = 3, 5, 6, 7, 8$.

(*ii*) $(f_2)_{[q,p,r,u,t,k]} = (f_1)_r$ for $k = 1, 2, 4$. For m in M, p in K, q in K_2, r in S_1 but not in F_1 and a in $I_2 - \{¢, \$\}$, we define g_2 by:

(*iii*) $g_2([q_2, p_0, r, \epsilon, 0, 1], ¢, m) = ([q_2, p_0, s, \epsilon, 0, 1], 0)$ if $g_1(r, ¢, m) = (s, 0)$. (A_2 simulates A_1, scanning and remaining at ¢ on its input.)

(*iv*) $g_2([q_2, p_0, r, \epsilon, 0, 1], ¢, m) = ([q_2, p_0, s, \epsilon, 0, 2], +1)$ if $g_1(r, ¢, m) = (s, +1)$. (A_2 simulates A_1 moving right from ¢. The pointer is set to 2, so A_2 will next compute the output of G for the symbol it will next scan on its input.)

(*v*) $g_2([q_2, p, r, \epsilon, 0, 1], \$, m) = ([q_2, p, s, \epsilon, 0, 1], 0)$ if $g_1(r, \$, m) = (s, 0)$. ($A_2$ simulates A_1 scanning and remaining at \$.)

(*vi*) $g_2([q_2, p, r, \epsilon, 0, 1], \$, m) = ([q_1, p, s, \epsilon, 0, 4], 0)$ if $g_1(r, \$, m) =$

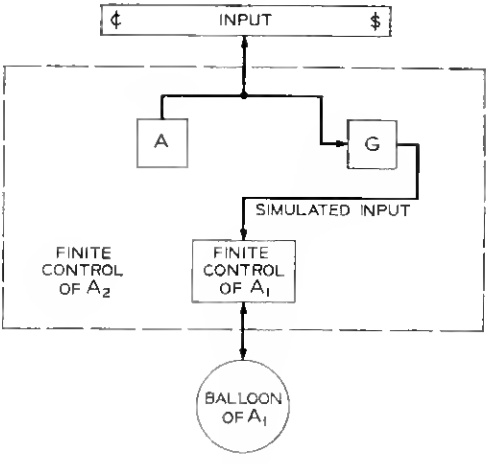

Fig. 3 — Automaton A_2.

$(s, -1)$. (A_2 simulates A_1 moving left from \$, and prepares to simulate A. The pointer is set to 4, and the first component to q_1.)

(*vii*) $g_2([q, p, r, \epsilon, 0, 4], a, m) = g_2([q, p, r, \epsilon, 0, 5], a, m) = ([q', p', r, \epsilon, 0, 5], d)$ if $\delta_1([q, p], a) = ([q', p'], d)$, for $q \neq q_2$, and a in I_2. (A_2 simulates A in A_2's first two components of state. The pointer is held at 5.)†

(*viii*) $g_2([q_2, p, r, \epsilon, 0, 5], a, m) = ([q_2, p, r, u, l, 6], 0)$ if $u = \lambda(p, a)$ and $u \neq \epsilon$. Here, $| u | = l$. (A_2 computes the output of G and prepares to simulate A_1. The pointer is set to 6.)

(*ix*) If instead, $\lambda(p, a) = \epsilon$, $g_2([q_2, p, r, \epsilon, 0, 5], a, m) = ([q_1, p, r, \epsilon, 0, 5], 0)$. ($A_2$ must simulate A again to find an input symbol that gives an output $\neq \epsilon$.)

(*x*) $g_2([q_1, p_0, r, \epsilon, 0, 4], \not{c}, m) = g_2([q_1, p_0, r, \epsilon, 0, 5], \not{c}, m)$ and is equal to $g_2([q_2, p_0, r, \epsilon, 0, 1], \not{c}, m)$ as defined by rules (*iii*) and (*iv*). (A_2 was prepared to begin simulating A, but found itself at the left endmarker. Note that in this case, the state of G must be p_0. A_2 immediately simulates A_1.)

(*xi*) $g_2([q_2, p, r, \epsilon, 0, 2], a, m) = g_2([q_2, p, r, \epsilon, 0, 7], a, m) = ([q_2, p, r, u, 1, 6], 0)$ if $\lambda(p, a) = u$ and $u \neq \epsilon$. (A_2 has simulated a move right of A_1's input head. It computes the output of G and prepares to simulate A_1. The pointer is set to 6, as in rule (*viii*).)

(*xii*) If instead, $\lambda(p, a) = \epsilon$, $g_2([q_2, p, r, \epsilon, 0, 2], a, m) = g_2([q_2, p, r, \epsilon, 0, 7], a, m) = ([q_2, t, r, \epsilon, 0, 7], +1)$ if $\delta(p, a) = t$. (A_2 must search right, in order to find an input symbol that does not give ϵ output when given to G.)

(*xiii*) $g_2([q_2, p, r, \epsilon, 0, 2], \$, m) = g_2([q_2, p, r, \epsilon, 0, 7], \$, m)$ and is equal to $g_2([q_2, p, r, \epsilon, 0, 1], \$, m)$ as defined by rules (*v*) and (*vi*). (A_2 was simulating a move by A_1, but encountered the right endmarker. A_2 immediately simulates another move of A_1.)

(*xiv*) Suppose $u \neq \epsilon$ and $1 \leqq l \leqq | u |$. Also, suppose $g_1(r, b, m) = (s, d)$, where b is the lth symbol of u, and $1 \leqq l + d \leqq | u |$. Then, $g_2([q_2, p, r, u, l, 1], a, m) = g_2([q_2, p, r, u, l, 6], a, m) = ([q_2, p, s, u, l + d, 1], 0)$. ($A_2$ simulates a move of A_1, where A_1 is assumed scanning the lth symbol of u.)

(*xv*) Under the assumptions of (*xiv*), if $l + d = 0$, $g_2([q_2, p, r, u, l, 1], a, m) = g_2([q_2, p, r, u, l, 6], a, m) = ([q_1, p, s, \epsilon, 0, 4], 0)$. ($A_2$ simulates A_1, but finds that A_1 moves left from u. A_2 prepares to simulate A.)

(*xvi*) Under the assumptions of (*xiv*), if $l + d > | u |$, $g_2([q_2, p, r,$

† Recall δ_1 is the next state mapping of A.

$u, l, 1], a, m) = g_2([q_2, p, r, u, l, 6], a, m) = ([q_2, t, s, \epsilon, 0, 2], +1)$, where $t = \delta(p, a)$. (A_2 simulates A_1, but finds that A_1 moves right from u. A_2 simulates the state transition of G.)

For r in F_1 and any k:

(*xvii*) $g_2([q, p, r, u, l, k], a, m) = ([q, p, r, u, l, 8], 0)$. ($A_1$ has been simulated entering an accepting state. A_2 sets the pointer to 8 and accepts.)

By rule (*xvii*) above, we see that exactly when A_2 gets to a state with third component in F_1 will it accept. It is sufficient to show that A_2 can simulate any single move of A_1 which does not start from an accepting state.

Formally, let us focus our attention on a particular input, $\not\!c w\$$, to A_2, where w is in $(I_2 - \{\not\!c, \$\})^*$. Let $G(w) = v$ and $| v |$ be n. For this particular w, and configuration $(r, \not\!c v\$, j, i)$ of A_1, we define the *inverse image* of $(r, \not\!c v\$, j, i)$, denoted $\mathrm{II}(r, \not\!c v\$, j, i)$ as follows:

(*i*) If $j = 0$, then $([q, p_0, r, \epsilon, 0, k], \not\!c w\$, 0, i)$ is in $\mathrm{II}(r, \not\!c v\$, 0, i)$ if either $k = 1$ and $q = q_2$ or $k = 4$ and $q = q_1$, or $k = 5$ and $q = q_1$.

(*ii*) If $j = n + 1$, then $([q_2, p, r, \epsilon, 0, k], \not\!c w\$, n_1 + 1, i)$ is in $\mathrm{II}(r, \not\!c v\$, n + 1, i)$ if $p = \delta(p_0, w)$ and $k = 1, 2$ or 7. Here $n_1 = | w |$.

(*iii*) If $1 \leqq j \leqq n$, $([q_2, p, r, u, l, k], \not\!c w\$, j_1, i)$ is in $\mathrm{II}(r, \not\!c v\$, j, i)$ if one can write $v = v_1 u v_2$ and $w = w_1 a w_2$, a in $I_2 - \{\not\!c, \$\}$, such that the following is true:

(*a*) $\delta(p_0, w_1) = p$
(*b*) $\lambda(p_0, w_1) = v_1$
(*c*) $\lambda(p, a) = u \neq \epsilon$
(*d*) $j_1 = | w_1 | + 1$
(*e*) $j = | v_1 | + l$
(*f*) $k = 1$ or 6.

Intuitively, A_2's input head is scanning the symbol giving rise, when fed to G, to the symbol scanned by the input head of A_1.

We must show that if $(r, \not\!c v\$, j_1, i_1) \vdash_{A_2} (s, \not\!c v\$, j_2, i_2)$, and r is not in F_1, then if $([q, p_1, r, u_1, l_1, k_1], \not\!c w\$, j_3, i_1)$ is a configuration in $\mathrm{II}(r_1, \not\!c v\$, j_1, i_1)$, then there is some configuration $([q', p_2, s, u_2, l_2, k_2], \not\!c w\$, j_4, i_2)$ in $\mathrm{II}(s, \not\!c v\$, j_2, i_2)$ such that:

$$([q, p_1, r, u_1, l_1, k_1], \not\!c w\$, j_3, j_1) \vdash^*_{A_2} ([q', p_2, s, u_2, l_2, k_2], \not\!c w\$, j_4, i_2).$$

Case 1: $j_1 = j_2 = 0$. The result follows trivially from rules (*ii*), (*iii*) and (*x*).

Case 2: $j_1 = j_2 = n + 1$. Trivial from rules (*ii*), (*v*) and (*xiii*).

Case 3: $j_1 = 0$, $j_2 = 1$. By rules (*ii*), (*iv*) and (*x*), $([q\,, p_0\,, r, \epsilon, 0, k], \not{c}w\$, 0, i_1) \vdash_{A_2} ([q_2\,, p_0\,, s, \epsilon, 0, 2], \not{c}w\$, 1, i_2)$. By rules (*i*), (*xi*) and (*xii*), if $v \neq \epsilon$, $([q_2\,, p_0\,, s, \epsilon, 0, 2], \not{c}w\$, 1, i_2) \vdash^*_{A_2} ([q_2\,, p, s, u, 1, 6], \not{c}w\$, j, i_2)$, where if $w = a_1a_2 \cdots a_{n_1}$, then $\delta(p_0\,, a_1a_2 \cdots a_{j-1}) = p$, $\lambda(p_0\,, a_1a_2 \cdots a_{j-1}) = \epsilon$ and $\lambda(p, a_j) = u$. If $v = \epsilon$, by rules (*i*) and (*xii*) $([q_2\,, p_0\,, s, \epsilon, 0, 2], \not{c}w\$, 1, i_2) \vdash^*_{A_2} ([q_2\,, p, s, \epsilon, 0, k_1], \not{c}w\$, n_1 + 1, i_2)$, where $p = \delta(p_0\,, w)$ and $k_1 = 2$ or 7.

Case 4: $j_1 = n + 1$, $j_2 = n$. By rules (*ii*), (*vi*) and (*xiii*), $([q_2\,, p_1\,, r, \epsilon, 0, k], \not{c}w,\$\, n_1 + 1, i_1) \vdash_{A_2} ([q_1\,, p_1\,, s, \epsilon, 0, 4], \not{c}w\$, n_1 + 1, i_2)$. If $v \neq \epsilon$, by Lemma 3, and rules (*i*), (*vii*), (*viii*) and (*ix*), $([q_1\,, p_1\,, s, \epsilon, 0, 4], \not{c}w\$, n_1\,, i_2) \vdash^*_{A_2} ([q_2\,, p_2\,, s, u, l, 6], \not{c}w\$\, j, i_2)$, where if $w = a_1a_2 \cdots a_{n_1}$, $\delta(p_0a_1a_2 \cdots a_{j-1}) = p_2$, $\lambda(p_0\,, a_1a_2 \cdots a_{j-1}) = v_1$, $v_1u = v$, and $\lambda(p_2\,, a_j) = u$. If $v = \epsilon$, by Lemma 3 and rules (*i*), (*vii*) and (*ix*), $([q_1\,, p_1\,, s, \epsilon, 0, 4], \not{c}w\$, n_1\,, i_2) \vdash^*_{A_2} ([q_1\,, p_0\,, s, \epsilon, 0, k], \not{c}w\$, 0, i_2)$, where $k = 4$ or 5.

Case 5: j_1 is not 0 or $n + 1$. Also, $l + j_2 - j_1$ lies between 1 and $|\, u \,|$, where l and u are defined in part (*iii*) of the definition of inverse image. The result is immediate from rules (*ii*) and (*xiv*).

Case 6: j_1 is not 0 or $n + 1$, but $l = |\, u \,|$ and $j_2 = j_1 + 1$. By rules (*ii*) and (*xvi*), $([q_2\,, p_1, r, u, l, k], \not{c}w\$, j_2\,, i_1) \vdash_{A_2} ([q_2\,, p_2\,, s, \epsilon, 0, 2], \not{c}w\$, j_3 + 1, i_2)$, where $([q_2\,, p_1\,, r, u, l, k], \not{c}w\$, j_3\,, i_1)$ is either of the inverse images of $(r, \not{c}v\$, j_1\,, i_1)$. The rest of the argument for this case is similar to that of case 3, and will be left to the reader.

Case 7: j_1 is not 0 or $n + 1$, but $l = 1$ and $j_2 = j_1 - 1$. By rules (*ii*) and (*xv*), $([q_2\,, p, r, u, l, k], \not{c}w\$, j_3\,, i_1) \vdash_{A_2} ([q_2\,, p, s, \epsilon, 0, 4], \not{c}w\$, j_3\,, i_2)$, where the former configuration is again either of the inverse images of $(r, \not{c}v\$, j_1\,, i_1)$. The argument proceeds as in case 4.

We claim, from the above, that $([q_2\,, p_0\,, r_1\,, \epsilon, 0, 1], \not{c}w\$, 0, 1) \vdash^*_{A_2} ([q, p, r, u, l, k], \not{c}w\$, j_1\,, i) \vdash_{A_2} ([q, p, r, u, l, 8], \not{c}w\$, j_1\,, i)$, for some r in F_1, $k \neq 8$, if and only if $(r_1\,, \not{c}v\$, 0, 1) \vdash^*_{A_1} (r, \not{c}v\$, j, 1)$ by a sequence of moves for which A_1 never previously enters an accepting state. Here u, l, j and j_1 are related as in part (*iii*) of the definition of inverse image. Thus, $T(A_2) = \{\not{c}w\$ \mid \text{for some } v \text{ with } \not{c}v\$ \text{ in } T(A_1),\ G(w) = v\}$. We must add that by Lemma 2, A_2 is in class C. The theorem is thus proven.

IV. OTHER TYPES OF BALLOON AUTOMATA

We have considered the two-way deterministic balloon automaton. To complete the story we should consider three other models—nondeterministic two-way balloon automata, and one-way balloon automata of the deterministic and non-deterministic varieties.

A nondeterministic device typically has the choice of a finite number of possibilities for each move. We choose to make the finite control function nondeterministic. This added capability enables us to represent the nondeterministic versions of the seven types of automata which we could represent by a deterministic balloon automaton.

A one-way balloon automaton is, quite naturally, a two-way balloon automaton, restricted so that the input head can only move right or not move at all.

We shall not repeat the definitions for each of the three new types of balloon automata, but, as a model, shall make use of the definition of two-way deterministic balloon automata.

A *two-way, nondeterministic balloon automaton* is denoted $A = (S, I, M, f, g, h, q_0, F)$ where all components are defined exactly as for the deterministic case, except that g is a mapping from $S \times I \times M$ to the subsets of $S \times \{-1, 0 + 1\}$.

A *one-way, deterministic balloon automaton* is denoted as are the two-way types, but g is a mapping from $S \times I \times M$ to $(S \times \{0, +1\}) \cup \{\varphi\}$.

A *one-way nondeterministic balloon automaton* is denoted as are the two-way types, but g is a mapping from $S \times I \times M$ to the subsets of $S \times \{0, +1\}$.

The closed classes of one way nondeterministic balloon automata are similar to the abstract families of acceptors in Ref. 21.

We shall use the abbreviations 2DBA, 2NBA, 1DBA, and 1NBA for, respectively, two-way deterministic, two-way nondeterministic, one-way deterministic and one-way nondeterministic balloon automata.

A *configuration* of any of the four types is denoted as for the 2DBA, (q, w, j, i), where, q is the state of finite control, w the input, j the input head position, and i the state of the balloon.

The possible moves of the 2NBA are determined as one would expect. One uses the balloon information function. Based on the value of that function, the input symbol at the position of the input head, and the state of finite control, one chooses a pair of next state of finite control and direction of input head, according to g. Then, based on the new state, the balloon control function is used.

Formally, if $A = (S, I, M, f, g, h, q_0, F)$ is a 2NBA, and (q_1, w, j_1, i_1)

and (q_2, w, j_2, i_2) are configurations of A, witb n the length of w, then we say (q_1, w, j_1, i_1) goes to (q_2, w, j_2, i_2) by a single *move*, denoted $(q_1, w, j_1, i_1) \vdash_A (q_2, w, j_2, i_2)$ exactly wben for some m in M, a in I, $d = -1$, 0 or $+1$, we have $h(i_1) = m$, the j_1th position of w is a, $g(q_1, a, m)$ contains (q_2, d) and $f_{a_2}(i_1) = i_2$. Also, $j_1 + d$ is between 0 and $n + 1$ and $j_2 = j_1 + d$. If (q_1, w, j_1, i_1) can go to configuration (q_2, w, j_2, i_2) by some number of moves, including 0, tben we say $(q_1, w, j_1, i_1) \vdash_A^* (q_2, w, j_2, i_2)$.

The notion of move, and the relations $\vdash$ and $\vdash^*$ are defined for the 1DBA and 1NBA exactly as for the 2DBA and 2NBA, respectively.

A 2NBA *accepts* an input, w if for some choice of moves it enters an accepting state. Formally, define $T(A)$, for a 2NBA, $A = (S, I, M, f, g, h, q_0, F)$ to be $\{w \mid (q_0, w, 0, 1) \vdash_A (q, w, j, i)$ for some q in $F\}$.

For the one-way types, we require that tbe input head reach the right endmarker when it accepts. That is, if $A = (S, I, M, f, g, h, q_0, F)$ is a 1NBA or 1DBA, then $T(A) = \{w \mid (q_0, w, 0, 1) \vdash_A^* (q, w, n + 1, i)$ for some q in F, wbere n is the length of $w\}$.

The notions of *closed class* of balloon automata for the 2NBA, 1DBA and 1NBA are defined exactly as for the 2DBA.

Note tbat, for example, a 1DBA is not a 1NBA, although there are obvious relationships. Also, strictly speaking, a closed class of 1DBA is not a closed class of 1NBA. Both parts I and II of the definition for 1NBA would require nondeterministic finite control functions in any class of 1NBA. Analogous statements hold between 2DBA and 2NBA, 1DBA and 2DBA, 1NBA and 2NBA.

It is trivial to see that Lemmas 1 and 2 hold for the 2NBA, 1DBA and 1NBA.

A set of languages is said to be a *closed class* (or simply *class*) for the 2NBA, 2DBA, 1NBA, or 1DBA if they are exactly the languages accepted by a closed class of automata of that type.

V. TWO-WAY NONDETERMINISTIC BALLOON AUTOMATA

Theorems 2, 3 and 4, proven for the 2DBA also hold for the 2NBA. In each case, the simulation by an automaton in some class, C, of one or two other automata in C was involved. In the 2NBA case, the simulation can be nondeterministic if the simulated automata are. We will therefore omit tbe proofs of tbe three theorems for the nondeterministic case.

Likewise, Tbeorem 1 holds for the 2NBA. We can simulate a 2NBA by a nondeterministic Turing machine just as we simulated the 2DBA

by a deterministic Turing machine. A nondeterministic Turing machine, as is well known, can be simulated by a deterministic Turing machine.

There is one additional, simple theorem we can prove for the 2NBA but not the 2DBA.

Theorem 5: If L_1 and L_2 are languages accepted by automata A_1 and A_2, respectively, in class C of 2NBA, *then there is an automaton, A_3, in C accepting $L_1 \cup L_2$.*

Proof: Let $A_1 = (S_1, I_1, M_1, f_1, g_1, h_1, q_1, F_1)$ and $A_2 = (S_2, I_2, M_2, f_2, g_2, h_2, q_2, F_2)$. As was mentioned, by Lemma 2 we can assume that $S_1 \cap S_2 = \varphi$ and $I_1 = I_2 = I$. Consider a new automaton, $A_3 = (S_3, I, M_3, f_3, g_3, h_3, q_3, F_3)$. $S_3 = S_1 \cup S_2 \cup \{q_3\}$, where q_3 is not in $S_1 \cup S_2$. $F_3 = F_1 \cup F_2$. $M_3 = M_1 \cdot M_2$ and $h_3 = h_1 \cdot h_2$.† Define f_3 by $(f_3)_{q_3} = \alpha^{(0)}$, $(f_3)_q = (f_1)_q$ if q is in S_1, and $(f_3)_q = (f_2)_q$ if q is in S_2.

Let the largest element of M_1 be k. We define g_3 as follows. For a in I and m in M_3, let $m_1 = \sigma_1(k, m)$ and $m_2 = \sigma_2(k, m)$. If q is in S_1, then $g_3(q, a, m) = g_1(q, a, m_1)$. If q is in S_2, $g_3(q, a, m) = g_2(q, a, m_2)$. Finally, $g_3(q_3, a, m) = g_1(q_1, a, m_1) \cup g_2(q_2, a, m_2)$.

It is straightforward to see that A_3 is in class C.

It should be clear that for any input, w, $(q_3, w, 0, 1) \vdash_{A_3} (q, w, j, i)$ exactly when either q is in S_1 and $(q_1, w, 0, 1) \vdash_{A_1} (q, w, j, i)$ or q is in S_2 and $(q_2, w, 0, 1) \vdash_{A_2} (q, w, j, i)$. Also, once in a state of S_1, A_3 remains in a state of S_1 and simulates A_1. Likewise, in a state of S_2, A_3 simulates A_2. Thus, by induction on the number of moves made, starting with one move, we have $(q_3, w, 0, 1) \vdash^*_{A_3} (q, w, j, i)$ if and only if $(q_1, w, 0, 1) \vdash^*_{A_1} (q, w, j, i)$ or $(q_2, w, 0, 1) \vdash^*_{A_2} (q, w, j, i)$. Thus, since $F_3 = F_1 \cup F_2$, and neither q_1 or q_2 may be in F_3, it follows that $T(A_3) = T(A_1) \cup T(A_2)$.

VI. ONE-WAY DETERMINISTIC BALLOON AUTOMATA

The 1DBA is the poorest of the four types in terms of the operations on languages which preserve membership in a closed class of languages for given types. Of the operations preserving membership in class for the two-way devices, only inverse g.s.m. mappings preserve membership in class for the 1DBA. The proof is along the lines of that of Theorem 4, but is simpler because the input head never has to move left. We will omit the proof.

There is one new operation which does preserve classes for the 1DBA, and, incidently, the 1NBA. This operation is intersection with

† Recall the definition of the operation "$\cdot$", σ_1 and σ_2 in Section III.

a regular set. Classes for the 2NBA and 2DBA were closed under intersection of languages in the class. A simple use of part I of the definition of closed class shows that every regular set is in every closed class, so intersection with a regular set surely preserves membership in class for the 2NBA and 2DBA.

We shall give the usual formal definition of a finite automaton. See Ref. 20, for example. A *finite automaton* is a 5-tuple, $A = (K, \Sigma, \delta, q_0, F)$. K is the finite set of *states*, Σ the finite set of *input symbols*. F is a subset of K, the *final states*, and q_0, in K is the *start state*. δ is a map from $K \times \Sigma$ to K. We extend δ to domain $K \times \Sigma^*$ by $\delta(q, \epsilon) = q$ for all q in K, and $\delta(q, wa)$, for q in K, w in Σ^* and a in Σ is $\delta(\delta(q, w), a)$. Define $T(A) = \{w \mid \delta(q_0, w) \text{ is in } F\}$. The finite automata accept exactly the regular sets.

Theorem 6: Let C be a class of one-way, deterministic balloon automata. Let L be accepted by some automaton, A in C, and let R be a regular set. Then $L \cap R$ is accepted by some automaton in class C.

Proof: Let $A = (S, I, M, f, g, h, q_0, F)$ be a 1DBA. Let $R_1 = R \cap \not{c}(I - \{\not{c}, \$\})^*\$$, and let $R_2 = \{w \mid w\$ \text{ is in } R_1\}$. If R is regular, then R_1 and R_2 are both regular. It is sufficient to show that there is an automaton in C accepting $L \cap R_1$. To that end, let $A_1 = (K, I, \delta, p_0, F_1)$ be a finite automaton with $T(A_1) = R_2$. Define $A_2 = (S_2, I, M, f_2, g_2, h, q_2, F_2)$ to be a 1DBA, with $S_2 = S \times K$, $q_2 = [q_0, p_0]$ and $F_2 = F \times F_1$. Define f_2 and g_2 as follows, for all q and q_1 in S, p and p_1 in K, a in I and m in M:

(*i*) Suppose $g(q, a, m) = (q_1, 0)$. Then for all p in K, $g_2([q, p], a, m) = ([q_1, p], 0)$.

(*ii*) Suppose $g(q, a, m) = (q_1, +1)$ and $\delta(p, a) = p_1$. Then $g_2([q, p], a, m) = ([q_1, p_1], +1)$.

(*iii*) $(f_2)_{[a,p]} = f_a$ for all p in K.

The states of A_2's finite control have two components. The first is a state of A and the second a state of A_1. By rules (*i*) and (*iii*), when the input head of A does not move, A_2 simulates a move of A, but does not change the state of A_1. By rules (*ii*) and (*iii*), when the input head of A moves right, A_2 simulates that move also, but adjusts the state of A_1 in the logical manner.

Formally, we can show by induction on the number of moves of A or A_2, starting with 0 moves, that $([q_0, p_0], w, 0, 1) \overset{*}{\underset{A_2}{\vdash}} ([q, p], w, j, i)$ if and only if:

(*i*) $(q_0, w, 0, 1) \overset{*}{\underset{A}{\vdash}} (q, w, j, i)$, and
(*ii*) $\delta(p_0, w_1) = p$, where w_1 is that portion of w to the left of position j.

Now a word, w, of length n, is accepted by A_2 if and only if $([q_0, p_0], w, 0, 1) \overset{*}{\underset{A_2}{\vdash}} ([q, p], w, n + 1, i)$, for some q in F, p in F_1 and any integer, i. The above is equivalent to saying that $(q_0, w, 0, 1) \overset{*}{\underset{A}{\vdash}} (q, w, n + 1, i)$ and $\delta(p_0, w_1) = p$, where $w_1\$ = w$. That is, w is in $T(A)$ and w_1 is in $T(A_1)$. But w_1 is in $R_2 = T(A_1)$ if and only if w is in R_1. Thus, $T(A_2) = L \cap R_1$. It should be clear, by Lemma 2, that A_2 is in class C.

Corollary 1: If L is a language in class C for the 1DBA, *and R is a regular set, then $L - R$ is in class C.*

Proof: Let L be contained in I^* for some finite alphabet, I. Then $L - R = L \cap (I^* - R)$, which is in class C by Theorem 6.

Theorem 6 applies also to the 1NBA. In fact, there is an additional corollary that can be shown for the 1NBA.

Corollary 2: Let L be in class C of 1NBA, *and let R be a regular set not involving symbols ¢ or \$. Then $L \cup ¢R\$$ is in class C.*

Proof: The results is a simple extension of Theorem 6, and will he left to the reader.

VII. ONE-WAY NONDETERMINISTIC BALLOON AUTOMATA

As the 1DBA was the poorest of the four models, in terms of provable properties, the 1NBA is the richest. Theorem 4, concerning inverse g.s.m. mappings, certainly holds for the 1NBA, as do Theorem 5, Theorem 6 and its corollaries.

To begin a study of the 1NBA, we will show that with the proper definition of acceptance, endmarkers on the input are not necessary. Let $A = (S, I, M, f, g, h, q_0, F)$ be a 1NBA. We informally define $\hat{T}(A)$ as the set of strings, w, in $(I - \{¢, \$\})^*$ which cause A to leave w moving right, at the same time entering an accepting state.

We need a slightly revised notion of a configuration. Since w has no endmarkers, its length is the number of symbols comprising w. (Recall, we never counted endmarkers in determining length.) Let w be of length n. Then (q, w, j, i) is a configuration of A if q is in S, i is an integer and $1 \leqq j \leqq n$. The initial configuration for a 1NBA without endmarkers is $(q_0, w, 1, 1)$. For convenience, we define a configuration, (*), which is imagined to result when A is in a configura-

tion (q, w, n, i), and the finite control function allows A, on the next move, to move its input head right and enter an accepting state.† There is no change in the definitions of $\vdash_A$ and $\vdash_A^*$. The former relates two configurations if the second is obtainable from the first by a single move, and the latter — if hy some finite number of moves. Note that no configuration can result from (*).

Now, we define $\hat{T}(A)$ as $\{w \mid (q_1, w, 1, 1) \vdash_A^* (*)\}$. When talking of a 1NBA and the $\hat{T}$ definition of acceptance, we will allow the start state to be an accepting state. If so, we shall, hy convention, say that ϵ is in $\hat{T}(A)$. We will endeavor to show that a language is $T(A_1)$ for some 1NBA, A_1, if and only if it is $\hat{T}(A_2)$ for A_2, a 1NBA in the same classes as A_1. The result is hroken into two parts.

Theorem 7: Let $A_1 = (S_1, I, M, f_1, g_1, h, q_1, F_1)$ *be a* 1NBA *with* $L = \hat{T}(A_1)$. *Then there is another* 1NBA, $A_2 = (S_2, I, M, f_2, g_2, h, q_2, F_2)$, *such that* $\cent L\$ = T(A_2)$.‡ *Moreover, if* A_1 *is in some closed class,* C, *then* A_2 *is in* C.

Proof: Choose q_2 to he a symhol not in S_1, and let $S_2 = S_1 \cup \{q_2\}$. $F_2 = F_1$ if q_1 is not in F_1; $F_2 = F_1 \cup \{q_2\}$ otherwise. Define f_2 and g_2 as follows:

(*i*) $(f_2)_{q_2} = \alpha^{(0)}$
(*ii*) $(f_2)_q = (f_1)_q$ for q in S_1.

For all a in $I - \{\cent, \$\}$ and all m in M:

(*iii*) $g_2(q, a, m) = g_1(q, a, m)$, for q in S_1.
(*iv*) $g_2(q_2, a, m) = g_1(q_1, a, m)$.
(*v*) $g_2(q, \$, m) = \varphi$, for q in S_2.
(*vi*) $g_2(q_2, \cent, m) = \{(q_2, +1)\}$.
(*vii*) $g_2(q, \cent, m) = \varphi$ for q in S_1.

Let w, of length $n \geqq 1$ he in $(I - \{\cent, \$\})^*$. By rules (*i*) and (*vi*), we have $(q_2, \cent w\$, 0, 1) \vdash_{A_2} (q_2, \cent w\$, 1, 1)$. By rules (*ii*) and (*iv*), it follows that $(q_2, \cent w\$, 1, 1) \vdash_{A_2} (q, \cent w\$, j, i)$ if and only if $(q_1, w, 1, 1) \vdash_{A_1} (q, w, j, i)$. Then, hy induction on the number of moves made, starting with one move, we see that $(q_2, \cent w\$, 1, 1) \vdash_{A_2}^* (q, \cent w\$, j, i)$ if and only if $(q_1, w, 1, 1) \vdash_{A_1}^* (q, w, j, i)$, for $j \leqq n$. Finally, by rules (*ii*) and (*iii*), if $(q, w, n, i) \vdash_A (*)$, then it must be that $g_1(q, a, m)$ con-

† Strictly speaking we require also that the balloon control function be defined for i and the new state if configuration (*) is to be entered.
‡ Recall that by convention, $\cent$ and \$ are in every set of input symbols, even if endmarkers are not used.

tains $(p, +1)$ for some p in F_1, where $h(i) = m$ and a is the nth symbol of w. Also, $f_p(i)$ is defined, so $(q, ¢w\$, n, i) \vdash_{A_1} (p, ¢w\$, n + 1, i_1)$, where $i_1 = f_p(i)$. Thus, if w is in $\hat{T}(A_1)$, then $¢w\$$ is in $T(A_2)$.

If $(q_1, ¢w\$, 0, 1) \vdash^*_{A_2} (p, ¢w\$, n + 1, i)$, where p is in F_1, from rule (v) we see that A_2 could not have made a move while scanning the \$. Thus, for some q in S and integer, k, $(q_1, ¢w\$, 0, 1) \vdash^*_{A_2} (q, ¢w\$, n, k) \vdash_{A_2} (p, ¢w\$, n + 1, i)$. From the previous paragraph, we know that $(q_1, w, 1, 1) \vdash^*_{A_1} (q, w, n, k)$ and $(q, w, n, k) \vdash_{A_1} (*)$. Thus, if $¢w\$$ is in $T(A_2)$, then w is in $\hat{T}(A_1)$.

One detail remains, concerning the case $w = \epsilon$. If ϵ is in $\hat{T}(A_1)$, then q_1 is in F_1. Thus, q_2 is in F_2. By rule (vi), $(q_2, ¢\$, 0, 1) \vdash_{A_2} (q_2, ¢\$, 1, 1)$, so $¢\$$ is in $T(A_2)$. If ϵ is not in $\hat{T}(A)_1$, then q_1 is not in F_1 and q_2 is not in F_2. By rules (v) and (vi), only one move of A_2 is possible, and A_2 does not accept $¢\$$. We conclude that $T(A_2) = ¢L\$$. It is clear from Lemma 2 that A_2 is in class C.

Theorem 8: Let $A_1 = (S_1, I, M, f_1, g_1, h, q_1, F_1)$ *be a* 1NBA, *in some closed class,* C, *with* $L_1 = T(A_1)$. *Let* $L_2 = \{w \mid ¢w\$ \text{ is in } L_1\}$. *Then there is a* 1NBA, $A_2 = (S_2, I, M, f_2, g_2, h, [q_1, 1], F_2)$ *in class* C, *with* $\hat{T}(A_2) = L_2$.

Proof: We will place in S_2 all symbols of the form $[q, i]$, where q is in S_1 and $i = 1, 2, 3$, or 4. If $¢\$$ is in L_1, then $F_2 = \{[q_1, 1]\} \cup \{[q, 4] \mid q \text{ in } S_1\}$. If $¢\$$ is not in L_1, $F_2 = \{[q, 4] \mid q \text{ in } S_1\}$.† Define f_2 and g_2 as follows:

(*i*) $(f_2)_{[q,i]} = (f_1)_q$ for all q in S_1 and $i = 1, 2, 3, 4$.

For all m in M, q in S_1 and a in $I - \{¢, \$\}$:

(*ii*) $g_2([q, 1], a, m) = \{([p, 1], 0) \mid (p, 0) \text{ is in } g_1(q, ¢, m)\} \cup \{([p, 2], 0) \mid (p, +1) \text{ is in } g_1(q, ¢, m)\}$.

(*iii*) $g_2([q, 2], a, m) = \{([p, 2], d) \mid (p, d) \text{ is in } g_1(q, a, m), d = 0 \text{ or } +1\} \cup \{([p, 3], 0) \mid (p, +1) \text{ is in } g_1(q, a, m)\} \cup \{([p, 4], +1) \mid (p, +1) \text{ is in } g_1(q, a, m) \text{ and } p \text{ is in } F_1\}$.

(*iv*) $g_2([q, 3], a, m) = \{([p, 3], 0) \mid (p, 0) \text{ is in } g_1(q, \$, m) \text{ and } p \text{ is not in } F_1\} \cup \{([p, 4], +1) \mid (p, 0) \text{ is in } g_1(q, \$, m) \text{ and } p \text{ is in } F_1\}$.

(*v*) $g_2([q, 4,], a, m) = \varphi$ for any a in I, including $¢$ and \$.

Intuitively, when the second component of state of A_2 is 1, A_2 imagines it is reading $¢$ on its input. If the second component is 3, it imagines

† Obviously, it may not be possible to tell whether $¢\$$ is in L_1. In that case, the procedure given here can be thought of as defining two automata, one of which accepts L_2. Computation of A_2 from A_1 is not effective, but this fact will not alter our theoretical results.

it is reading \$. If the second component is 2, it uses the symbol actually scanned. A second component of 4 indicates an accepting state.

Formally, let w be in $(I - \{\not\epsilon, \$\})^*$, of length $n \geqq 1$. From rules (i) and (ii), we see that $([q_1, 1], w, 1, 1) \overset{*}{\underset{A_1}{\vdash}} ([p, 1], w, 1, i)$ if and only if $(q_1, \not\epsilon w\$, 0, 1) \overset{*}{\underset{A_1}{\vdash}} (p, \not\epsilon w\$, 0, i)$. Also, $([q, 1], w, 1, i_1) \underset{A_1}{\vdash} ([p, 2], w, 1, i_2)$ if and only if $(q, \not\epsilon w\$, 0, i_1) \underset{A_1}{\vdash} (p, \not\epsilon w\$, 1, i_2)$.

Next, by rules (i) and (iii), $([p, 2], w, 1, i_1) \overset{*}{\underset{A_1}{\vdash}} ([p, 2], w, n, i_2)$ if and only if $(q, \not\epsilon w\$, 1, i_1) \overset{*}{\underset{A_1}{\vdash}} (p, \not\epsilon w\$, n, i_2)$. Also, $([q, 2], w, n, i_1) \underset{A_1}{\vdash} ([p, 3], w, n, i_2)$ if and only if $(q, \not\epsilon w\$, n, i_1) \underset{A_1}{\vdash} (p, \not\epsilon w\$, n + 1, i_2)$. In addition, $([q, 2], w, n, i_1) \underset{A_1}{\vdash} (*)$ if and only if $(q, \not\epsilon w\$, n, i_1) \underset{A_1}{\vdash} (p, \not\epsilon w\$, n + 1, i_1)$ for some p in F_1. For in the latter case, $([p, 4], +1)$ will be in $g_2(q, a, m)$, where $m = h(i_1)$ and a is the nth symbol of w. Note that $(f_2)_{[p,4]}$ is defined exactly when $(f_1)_p$ is defined.

Third, by rules (i) and (iv), $([q, 3], w, n, i_1) \overset{*}{\underset{A_1}{\vdash}} ([p, 3], w, n, i_2)$ if and only if $(q, \not\epsilon w\$, n + 1, i_1) \overset{*}{\underset{A_1}{\vdash}} (p, \not\epsilon w\$, n + 1, i_2)$ by a sequence of moves such that A_1 does not enter a state of F_1. Also, $([q, 3], w, n, i_1) \underset{A_1}{\vdash} (*)$ if and only if $(q, \not\epsilon w\$, n + 1, i_1) \underset{A_1}{\vdash} (p, \not\epsilon w\$, n + 1, i_2)$ for some p in F_1.

Putting together the results above, we have that for w of length $n \geqq 1$, $([q_1, 1], w, 1, 1) \overset{*}{\underset{A_1}{\vdash}} (*)$ if and only if $(q_1, \not\epsilon w\$, 0, 1) \overset{*}{\underset{A_1}{\vdash}} (p, \not\epsilon w\$, n + 1, i)$ for some p in F_1 and integer, i. Also, $[q_1, 1]$ is in F_2 if and only if $\not\epsilon\$$ is in L_1. Thus, ϵ is in $\hat{T}(A_2)$ exactly when $\not\epsilon\$$ is in $T(A_1)$. We conclude $\hat{T}(A_2) = L_2$. It is again straightforward to see that A_2 is in class C.

We say a closed class of automata is *recursive* if there is an algorithm to determine if any given word is in $T(A)$, for any automaton A in the class. We have a corollary to Theorem 8.

Corollary: If C is recursive, then for L_1 and L_2 as in Theorem 8, we can effectively find an automaton, A_2, with $\hat{T}(A_2) = L_2$, from the specification for A_1.

Proof: It is sufficient to note that in this case, we can effectively determine if $\not\epsilon\$$ is in L_1, hence we can effectively find A_2.

We can now prove a series of closure properties of the 1NBA.

Theorem 9: Let A_1 and A_2 be 1NBA *in some closed class, C. Let $L_1 = \hat{T}(A_1)$ and $L_2 = \hat{T}(A_2)$. Then there exists A_6 in C with $\hat{T}(A_6) = L_1L_2 = \{w \mid w = uv$ and u is in L_1, v in $L_2\}$.*

Proof: Let $A_1 = (S_1, I, M_1, f_1, g_1, h_1, q_1, F_1)$ and $A_2 = (S_2, I, M_2, f_2, g_2, h_2, q_2, F_2)$. By Lemma 2, we can assume that S_1 and S_2 are disjoint, and L_1 and L_2 are contained in $(I - \{\not\epsilon, \$\})^*$. Define

$A_3 = (S_3, I, M_3, f_3, g_3, h_3, [q_1, 1], F_2)$, where $h_3 = h_1 \cdot h_2$, $M_3 = M_1 \cdot M_2$, and $S_3 = S_2 \cup \{q_3\} \cup \{[q, i] \mid q \text{ in } S_1, i = 1 \text{ or } 2\}$, where q_3 is a new symbol. $\hat{T}(A_3)$ will not be L_1L_2, but rather $\{w | w = uv, u \text{ and } v \neq \epsilon, u \text{ in } L_1, v \text{ in } L_2\}$. We define f_3 and g_3 as follows:

(*i*) $(f_3)_{[q,i]} = (f_1)_q$ if q is in S_1, $i = 1$ or 2.
(*ii*) $(f_3)_q = (f_2)_q$ if q is in S_2.
(*iii*) $(f_3)_{q_3} = \alpha^{(1)}$.

For all a in $I - \{¢, \$\}$ and m in M_3, with k the largest element in M_1:

(*iv*) If q is in S_1 but not in F_1, then $g_3([q, i], a, m) = \{([p, 1], 0) \mid (p, 0)$ is in $g_1(q, a, \sigma_1(k, m))\} \cup \{([p, 2], +1) \mid (p, +1)$ is in $g_1(q, a, \sigma_1(k, m))\}$, $i = 1$ or 2.

(*v*) If q is in F_1, then $g_3([q, 1], a, m)$ is defined as in (*iv*). $g_3([q, 2], a, m) = \{([p, 1], 0) \mid (p, 0)$ is in $g_1(q, a, \sigma_1(k, m))\} \cup \{([p, 2], +1) \mid (p, +1)$ is in $g_1(g, a, \sigma_1(k, m))\} \cup \{(q_3, 0)\}$.

(*vi*) $g_3(q_3, a, m) = g_2(q_2, a, \sigma_2(k, m))$.

(*vii*) $g_3(q, a, m) = g_2(q, a, \sigma_2(k, m))$ for all q in S_2.

Note that if A_3 is in a state of the form $[q, 2]$, then on its last move, its input head moved right. If in a state of the form $[q, 1]$, the input head did not move right on the previous move. When A_3 has just moved right and entered an accepting state, according to rule (*v*), it has the option of continuing to simulate A_1 or going to state q_3, resetting the balloon to state 1, and then simulating A_2.

Formally, from rules (*i*), (*iv*) and (*v*), it is straightforward to show that $([q_1, 1], w, 1, 1) \overset{*}{\underset{A_3}{\vdash}} ([p, 2], w, j, i)$ for $j \geq 2$ and p in F, if and only if $(q_1, u, 1, 1) \overset{*}{\underset{A_1}{\vdash}}$ (*), where u is the first $j - 1$ symbols of w. Certainly, if and only if p is in F_1 does $([p, 2], w, j, i) \underset{A_3}{\vdash} (q_3, w, j, 1)$, by rules (*iii*) and (*v*). Finally, by rules (*ii*), (*vi*) and (*vii*), $(q_3, w, j, 1) \overset{*}{\underset{A_3}{\vdash}}$ (*) if and only if $(q_2, v, 1, 1) \overset{*}{\underset{A_2}{\vdash}}$ (*), where v is the $j$tb and subsequent symbols of w.

Thus, $\hat{T}(A_3) = \{w \mid w = uv$ with $u, v \neq \epsilon, u$ in L_1, v in $L_2\}$. Clearly, A_3 is in C. Suppose ϵ is in L_1. By Theorem 5, redone for the 1NBA, and Theorems 7 and 8, there exists A_4 in C with $\hat{T}(A_4) = \hat{T}(A_3) \cup L_2$. If ϵ is not in L_1, let $A_4 = A_3$. If ϵ is in L_2, there exists A_5 in C with $\hat{T}(A_5) = \hat{T}(A_4) \cup L_1$. If ϵ is not in L_2, let $A_5 = A_4$. Finally, if ϵ is in both L_1 and L_2, by Corollary 2 to Theorem 6, redone for the 1NBA, and Theorems 7 and 8, there exists A_6 in C with $\hat{T}(A_6) = \hat{T}(A_5) \cup \{\epsilon\}$. Otherwise, let $A_6 = A_5$. In any case, it should be clear that $\hat{T}(A_6) = L_1L_2$.

Corollary: If C is recursive, A_6 can be effectively found.

Proof: Immediate from the corollary to Theorem 8.

Theorem 10: Let A be a 1NBA *in class C, with $\hat{T}(A) = L$. Then there is an automaton, A_3, in C, with $\hat{T}(A_3) = L^* = \{\epsilon\} \cup L \cup LL \cup LLL \cup \cdots$.*

Proof: By Corollary 1 to Theorem 6, redone for 1NBA, and Theorems 7 and 8, there exists A_1 in C with $\hat{T}(A_1) = L_1 = L - \{\epsilon\}$. Note that $L^* = L_1^*$. Moreover, since ϵ is not in $\hat{T}(A_1)$, we can always effectively find A_1. Let $A_1 = (S_1, I, M, f_1, g_1, h, q_1, F_1)$. We will construct $A_2 = (S_2, I, M, f_2, g_2, h, q_2, F_2)$ in C, with $\hat{T}(A_2) = L_1^* - \{\epsilon\}$. Define $S_2 = \{q_2\} \cup \{[q, i] \mid q \text{ in } S_1, i = 1 \text{ or } 2\}$, where q_2 is a symbol not in S_1. Let $F_2 = \{[q, 2] \mid q \text{ in } F_1\}$. Define f_2 and g_2 as follows:

(*i*) $(f_2)_{[q,i]} = (f_1)_q$ for q in S_1, $i = 1$ or 2.
(*ii*) $(f_2)_{q_2} = \alpha^{(1)}$.
For all a in $I - \{\not\!\!\$, \$\}$ and m in M:
(*iii*) If q is in $S_1 - F_1$, $g_2([q, i], a, m) = \{([p, 1], 0) \mid (p, 0) \text{ is in } g_1(q, a, m)\} \cup \{([p, 2], +1) \mid (p, +1) \text{ is in } g_1(q, a, m)\}$, $i = 1$ or 2.
(*iv*) If q is in F_1, $g_2([q, 1], a, m)$ is as in rule (*iii*). $g_2([q, 2], a, m) = \{([p, 1], 0) \mid (p, 0) \text{ is in } g_1(q, a, m)\} \cup \{([p, 2], +1) \mid (p, +1) \text{ is in } g_1(q, a, m)\} \cup \{(q_2, 0)\}$.
(*v*) $g_2(q_2, a, m) = g_1(q_1, a, m)$.

The significance of 1 and 2 in the second component of state of S_2 is as in Theorem 9. A_2 simulates A_1, but when in an accepting state, just having moved its input head right, has the option of entering state q_2. Thus, it is easy to see that $(q_2, w, j_1, 1) \overset{*}{\underset{A_2}{\vdash}} ([p, 2], w, j_2, i)$, with $j_2 > j_1$ if and only if $(q_1, u, 1, 1) \overset{*}{\underset{A_1}{\vdash}} (*)$, where u is symbols j_1 through $j_2 - 1$ of w. Exactly when p is in F_1 do we have $([p, 2], w, j_2, i) \overset{-}{\underset{A_2}{\vdash}} (q_2, w, j_2, 1)$. Thus, if and only if $(q_2, w, 1, 1) \overset{*}{\underset{A_2}{\vdash}} (*)$, can w be written in the form $u_1 u_2 \cdots u_k$, $k \geqq 1$, where u_i, $1 \leqq i \leqq k$, is in L_1.

Thus, $\hat{T}(A_2) = L_1^* - \{\epsilon\} = L^* - \{\epsilon\}$. Surely, A_2 is in class C. By an argument used in Theorem 9, we can find A_3 in C, with $\hat{T}(A_3) = \hat{T}(A_2) \cup \{\epsilon\} = L^*$. Moreover, since ϵ is in $\hat{T}(A_3)$, we can always effectively find A_3.

Theorem 11: Let $A = (S, I, M, f, g, h, q_0, F)$ be a 1NBA *in class C. Let $G = (K, I - \{\not\!\!\$, \$\}, I_1 - \{\not\!\!\$, \$\}, \delta, \lambda, p_0)$ be a g.s.m. We assume for convenience that I_1 is a finite alphabet containing $\not\!\!\$$ and $\$$. Let $L = \{w \mid \not\!\!\$w\$ \text{ is in } T(A)\}$. Then there is an automaton, A_1 in C, with $T(A_1) = \not\!\!\$L_1\$$ and $L_1 = G(L)$.*

Proof: Let $A_1 = (S_1, I_1, M, f_1, g_1, h, q_1, F_1)$. The proof is represented in Fig. 4. The finite control of A_1 contains a generator which nondeterministically generates symbols in $I - \{\dot{\cent}, \$\}$. These symbols are processed by G, and compared with the input. The input head rests on the leftmost uncompared symbol. A_1 also uses the generated symbols as inputs to A, which it simulates. A_1 accepts if A accepts while A_1 is seanning \$ on the input, with no symhols left to compare.

We define $S_1 = \{[q, p, a, u, i] \mid q$ in S, p in K, a in $I - \{\$\}$ or $a = \epsilon$, $i = 1, 2$ or 3 and u in $(I - \{\dot{\cent}, \$\})^*$, but $\mid u \mid \leqq \max (\mid \lambda(p, a) \mid$ for p in K, a in $I.\}$. The first component keeps track of the state of A, the second, of the state of G. The third component holds the symbol generated, and the fourth, the output of G for that symbol and the current state of G. The last component is 1 usually. It is 2 when A would have just moved its input head right, and it is 3 when A would be scanning $\dot{\cent}$ or \$.

Define $F_1 = \{[q, p, \epsilon, \epsilon, i)] \, q$ in F, $i = 1$ or $3\}$. Also, $q_1 = [q_0, p_0, \epsilon, \epsilon, 3]$.

Define f_1 by:

(*i*) $(f_1)_{[q,p,a,u,i]} = f_q$ if either $a \neq \epsilon$ or $i = 3$.
(*ii*) $(f_1)_{[q,p,a,u,i]} = \alpha^{(0)}$ otherwise.

For all m in M, b in $I_1 - \{\dot{\cent}, \$\}$, q in S and p in K, define:

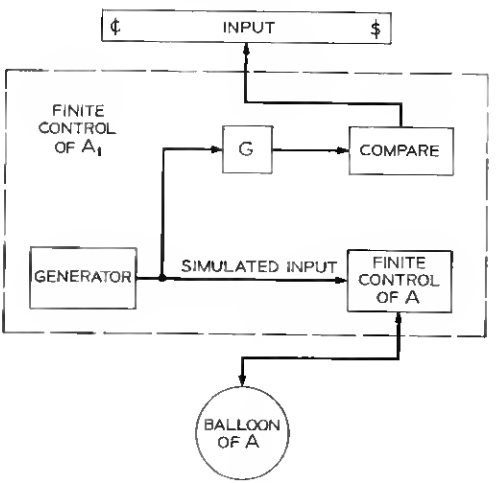

Fig. 4 — Automaton A_1.

(*iii*) $g_1([q, p_0, \epsilon, \epsilon, 3], \not\$, m)$ contains $([s, p_0, \epsilon, \epsilon, 3], 0)$ if $(s, 0)$ is in $g(q, \not\$, m)$.

(*iv*) $g_1([q, p_0, \epsilon, \epsilon, 3], \not\$, m)$ contains $([s, p_0, \not\$, \epsilon, 2], +1)$ if $(s, +1)$ is in $g(q, \not\$, m)$. ($A_1$ simulates A with $\not\$$ as input.)

(*v*) $g_1([q, p, \epsilon, \epsilon, 1], b, m)$ contains $\{([s, p_1, a, u, 1], 0) \mid$ for any a in $I - \{\not\$, \$\}$, if $(s, 0)$ is in $g(q, a, m)$, $p_1 = \delta(p, a)$, and $u = \lambda(p, a)\}$. Likewise, if $b = \$$.

(*vi*) $g_1([q, p, \epsilon, \epsilon, 1], b, m)$ contains $\{([s, p_1, a, u, 2], 0) \mid$ for a in $I - \{\not\$, \$\}$, if $(s, +1)$ is in $g(q, a, m)$, $p_1 = \delta(p, a)$, and $u = \lambda(p, a)\}$. Likewise, if $b = \$$. (The random generator generates symbol a, which is stored in the third component. $\lambda(p, a)$ is stored in the fourth. The new state of A, with a as input symbol is stored in the first component, and the new state of G in the second. If A would immediately move its input head right, the fifth component is 2. A 2 there tells A_1 it is finished with symbol a. Otherwise, a 1 is placed in the fifth component.)

(*vii*) $g_1([q, p, a, u, 1], b, m)$ includes $\{([s, p, a, u, 1], 0) \mid (s, 0)$ is in $g(q, a, m)\}$. Likewise, if $b = \$$.

(*viii*) $g_1([q, p, a, u, 1], b, m)$ includes $\{([s, p, a, u, 2], 0) \mid (s, +1)$ is in $g(q, a, m)\}$. Likewise, if $b = \$$. (A_1 simulates a move of A. The fifth component of A_1's state becomes 2 if the input head of A moves right.)

(*ix*) $g_1([q, p, a, u, 2], b, m) = \{([q, p, \epsilon, u, 1], 0)\}$. Likewise, if $b = \$$. (Remove a as third component and set fifth component to 1.)

(*x*) For any u, $g_1([q, p, \epsilon, bu, 1], b, m) = \{([q, p, \epsilon, u, 1], +1)\}$.

(*xi*) $g_1([q, p, \epsilon, bu, 1], b_1, m) = \varphi$ for $b_1 \neq b$. (A_1 compares its fourth component with the input.)

(*xii*) $g_1([q, p, \epsilon, \epsilon, 1], \$, m)$ contains $([s, p, \epsilon, \epsilon, 3], 0)$ if $g(q, \$, m)$ contains $(s, 0)$.

(*xiii*) $g_1([q, p, \epsilon, \epsilon, 3], \$, m)$ contains $([s, p, \epsilon, \epsilon, 3], 0)$ if $g(q, \$, m)$ contains $(s, 0)$.

We will state a series of intermediate results that follow directly from the rules given. We assume that $G(w) = x$.

I. By rules (*i*) and (*iii*): $([q_0, p_0, \epsilon, \epsilon, 3], \not\$x\$, 0, 1) \overset{*}{\underset{A_1}{\vdash}} ([q, p_0, \epsilon, \epsilon, 3], \not\$x\$, 0, i)$ if and only if $(q_0, \not\$w\$, 0, 1) \overset{*}{\underset{A}{\vdash}} (q, \not\$w\$, 0, i)$.

II. By rules (*i*), (*ii*), (*iv*), and (*ix*): $([q, p_0, \epsilon, \epsilon, 3], \not\$x\$, 0, i_1) \underset{A_1}{\vdash} ([s, p_0, \not\$, \epsilon, 2], \not\$x\$, 1, i_2) \underset{A_1}{\vdash} ([s, p_0, \not\$, \epsilon, \epsilon, 1], \not\$x\$, 1, i_2)$ if and only if $(q, \not\$w\$, 0, i_1) \underset{A}{\vdash} (s, \not\$w\$, 1, i_2)$.

III. By rules (*i*), (*v*) and (*vii*): $([q, p, \epsilon, \epsilon, 1], \not\$x\$, j, i_1) \overset{*}{\underset{A_1}{\vdash}} ([s, r, a, u, 1], \not\$x\$, j, i_2)$ by a sequence of moves for which the third component of state never becomes ϵ, or the fifth $= 2$, if and only if $(q, \not\$w\$, k, i_1)$

$\vdash_A^*$ $(s, \not\epsilon w\$, k, i_2)$, where the kth symbol of w is a, $\delta(p, a) = r$ and $\lambda(p, a) = u$.

IV. From III, and rules (i), (vi) and $(viii)$: $([q, p, \epsilon, \epsilon, 1], \not\epsilon x\$, j, i_1) \vdash_{A_1}^* ([s, r, a, u, 2], \not\epsilon x\$, j, i_2)$, by a sequence of moves in which the third component of state never becomes ϵ if and only if $(q, \not\epsilon w\$, k, i_1) \vdash_A^* (s, \not\epsilon w\$, k + 1, i_2)$ by a sequence of moves in whicb the input head remains at position k until the last move is made. Here, again, tbe kth symbol of w is a, $\delta(p, a) = r$ and $\lambda(p, a) = u$.

V. From rule (ix), $([q, p, a, u, 2], \not\epsilon x\$, j, i) \vdash_{A_1} ([q, p, \epsilon, u, 1], \not\epsilon x\$, j, i)$ for any $a \neq \epsilon$, and if the fifth component is 2, no other move is possible.

VI. From rule (x), $([q, p, \epsilon, u, 1], \not\epsilon x\$, j_1, i) \vdash_{A_1}^* ([q, p, \epsilon, \epsilon, 1], \not\epsilon x\$, j_2, i)$ by a sequence of moves for which the third component of state remains ϵ, if and only if symbols j_1 through $j_2 - 1$ of x form u.

VII. Combining IV, V, and VI: $([q, p, \epsilon, \epsilon, 1], \not\epsilon x\$, j_1, i_1) \vdash_{A_1}^* ([s, r, \epsilon, \epsilon, 1], \not\epsilon x\$, j_2, i_2)$ by a sequence of moves in which the third component changes from ϵ to a symbol in $I - \{\not\epsilon, \$\}$ back to ϵ only once, if and only if for some a in $I - \{\not\epsilon, \$\}$ and u in $(I_1 - \{\not\epsilon, \$\})^*$, we have

(a) $\delta(p, a) = r$;

(b) $\lambda(p, a) = u$;

(c) Symbols j_1 through $j_2 - 1$ of x are u;

(d) $(q, \not\epsilon w\$, k, i_1) \vdash_A^* (s, \not\epsilon w\$, k + 1, i_2)$ by a sequence of moves in which A's input head remains stationary until the last move, and a is the kth symbol of w.

Note that $j_1 = j_2 = n + 1$ is not prohibited.

VIII. Using I, II, and VII iterated: $([q_0, p_0, \epsilon, \epsilon, 3], \not\epsilon x\$, 0, 1) \vdash_{A_1}^* ([q, p, \epsilon, \epsilon, 1], \not\epsilon x\$, n + 1, i)$, where $| x | = n$, if and only if, for some w in $(I - \{\not\epsilon, \$\})^*$, of length k:

(a) $(q_0, \not\epsilon w\$, 0, 1) \vdash_A^* (q, \not\epsilon w\$, k + 1, i)$ by a sequence of moves in which A's input head does not reach $\$$ until the last move;

(b) $\delta(p_0, w) = p$;

(c) $\lambda(p_0, w) = x$.

IX. Directly from VIII, A_1 accepts $\not\epsilon x\$$ by entering a state $[q, p, \epsilon, \epsilon, 1]$, where q is in F, if and only if there is a w as in VIII such that A accepts $\not\epsilon w\$$ by entering state q on the same move on which A first moves its input head to $\$$.

X. From rules (xii) and $(xiii)$, $([q, p, \epsilon, \epsilon, 1], \not\epsilon w\$, n + 1, i_1) \vdash_{A_1}^* ([s, p, \epsilon, \epsilon, 3], \not\epsilon x\$, n + 1, i_2)$, where $| x | = n$, by a sequence of moves in which the third component of state remains ϵ, if and only if $(q, \not\epsilon w\$, k + 1, i_1) \vdash_A^* (s, \not\epsilon w\$, k + 1, i_2)$, where $k = | w |$.

XI. From VIII and X, A_1 accepts $\not\epsilon x\$$ by entering a state $[s, p, \epsilon, \epsilon, 3]$ if and only if A accepts $\not\epsilon w\$$, where $\lambda(p_0, w) = x$, by a sequence of

moves in which A enters state s while its input head remains scanning \$.

XII. Finally, from IX and XI, we have that A_1 accepts $\not\in x\$$ if and only if A accepts $\not\in w\$$, where $\lambda(p_0, w) = x$. Thus, $T(A_1) = \not\in G(L)\$$.

We need only add that A_1 is, by definition, in class C.

Corollary: If $L = \hat{T}(A)$ for some 1NBA, A, *in class C, and G is a g.s.m., then there is an automaton A_1 in class C, for which $\hat{T}(A_1) = G(L)$. If class C is recursive, we can effectively find A_1.*

Proof: Direct from Theorems 7 and 8.

Theorem 12: Let $L = \hat{T}(A)$ for some 1NBA, A, *in class C. Let R be a regular set. Then there is an automaton, A_3, in class C, with $\hat{T}(A_3) = L/R = \{w \mid$ for some x in R, wx is in $L\}$.*

Proof: Let $A = (S, I, M, f, g, h, q_0, F)$. We can surely find a finite automaton, $A_1 = (K, I - \{\not\in, \$\}, \delta, p_0, F_1)$ accepting $R \cap (I - \{\not\in, \$\})^*$. Intuitively, A_3 will simulate A, but will always have the additional choice of guessing that it has seen w. It then nondeterministically chooses the symbols of x, continuing to simulate A. We will construct A_2 to accept $L/R - \{\epsilon\}$. The reader can easily see how ϵ can be added to the set accepted by A_2.

Formally, let $A_2 = (S_2, I, M, f_2, g_2, h, q_0, F_2)$. $S_2 = S \cup \{[q, p, a] \mid q$ in S, p in K, a in $I - \{\not\in, \$\}$ or $a = \epsilon\}$. $F_2 = \{[q, p, \epsilon] \mid q$ in F, p in $F_1\}$. Define f_2 and g_2 as follows:

(*i*) $(f_2)_q = f_q$ for q in S.

(*ii*) $(f_2)_{[q,p,a]} = f_a$ for all q in S, p in K, a in $I - \{\not\in, \$\}$ or $a = \epsilon$.

For all b in $I - \{\not\in, \$\}$, m in M:

(*iii*) $g_2(q, b, m)$ contains (s, d) if $g(q, b, m)$ contains (s, d).

(*iv*) $g_2(q, b, m)$ contains $([s, p_0, \epsilon], +1)$ if $g(q, b, m)$ contains $(s, +1)$.

(*v*) $g_2(q, b, m)$ contains $([s, p, a], 0)$ if $g(q, b, m)$ contains $(s, +1)$ and $\delta(p_0, a) = p$, for any a in $I - \{\not\in, \$\}$.

(*vi*) $g_2([q, p, a], b, m)$ contains $([s, p, a], 0)$ if $g(q, a, m)$ contains $(s, 0)$.

(*vii*) $g_2([q, p, a], b, m)$ contains $([s, r, a_1], 0)$ for any a_1 in $I - \{\not\in, \$\}$ if $g(q, a, m)$ contains $(s, +1)$ and $\delta(p, a_1) = r$.

(*viii*) $g_2([q, p, a], b, m)$ contains $([s, p, \epsilon], +1)$ if $g(q, a, m)$ contains $(s, +1)$.

Note that no moves are possible if the third component of A_2's state is ϵ. From rules (*i*) and (*iii*), we see that $(q_0, w, 1, 1) \overset{*}{\underset{A_2}{\vdash}} (q, w, j, i)$ if and only if $(q_0, w, 1, 1) \overset{*}{\underset{A}{\vdash}} (q, w, j, i)$. Let w be of length n. By rules (*ii*) and (*iv*), $(q, w, n, i) \underset{A_2}{\vdash} (*)$ if and only if p_0 is in F_1 (i.e., ϵ is in R) and $(q, w, n, i) \underset{A}{\vdash} (*)$.

By rules (*ii*) and (*v*), $(q, w, n, i) \vdash_{A_3} ([s, p, a], w, n, i_2)$ if and only if $(q, wx, n, i_1) \vdash_A (s, wx, n + 1, i_2)$, where the first symbol of x is a and $\delta(p_0, a) = p$.† Let $| x | = k$. Then, by rules (*ii*), (*vi*), and (*vii*), $([q, p, a], w, n, i_1) \vdash^*_{A_3} ([s, r, a_1], w, n, i_2)$ if and only if $(q, wx, n + 1, i_1) \vdash^*_A (s, wx, n + k, i_2)$, $\delta(p, x) = r$, and x ends with a_1.

By rules (*ii*) and (*viii*), $([q, p, a], w, n, i) \vdash_{A_3} (*)$ if and only if $(q, wx, n + k, i) \vdash_A (*)$, where the last symbol of x is a, and x is in R.

Putting the above together, we see that $(q_0, w, 1, 1) \vdash^*_{A_3} (*)$ without ever entering a state of the form $[q, p, a]$, $a \neq \epsilon$ if and only if ϵ is in R and $(q_0, w, 1, 1) \vdash^*_A (*)$. Also $(q_0, w, 1, 1) \vdash^*_{A_3} (*)$, entering a state $[q, p, a]$, $a \neq \epsilon$ in so doing, if and only if for some x in $(I - \{¢, \$\})^*$, $(q_1, wx, 1, 1) \vdash^*_A (*)$ and x is in $R - \{\epsilon\}$. If A_2 is modified to accept ϵ, provided ϵ is in L/R, then the resulting device is A_3.

VIII. CONCLUSIONS

We have considered four types of general automata, and defined closed classes for each of these four types. We have shown certain common operations to preserve these classes, in the sense that if a language, L, is accepted by an automaton in the class, and L_1 is the result of the operation applied to L, then L_1 is accepted by some automaton in the class.

The classes model many of the common devices which have been heretofore considered in the literature, such as stack automata and counter machines. It seems as though they could be expected to model any future class of automata which are defined solely by the ways in which their infinite storage can be locally manipulated. The classes do not model such things as linear bounded automata or time/tape complexity classes of Turing machines, intuitively because such automata are defined by global restrictions on memory. (I.e., one may use "this much" memory, and no more.)

In Table I, we list the types of balloon automata and the operations considered. A check indicates that the operation preserves membership in a closed class of automata.

It is hoped that when models of automata are proposed in the future, theorists will find it efficient to show that their model is equivalent to a closed class of balloon automata. They will then have a variety of standard theorems already proven for them.

---

† Note, however, that the first symbol of x does not affect the operation of A at this step.

TABLE I

	2DBA	2NBA	1DBA	1NBA
Reversal	√	√		
Intersection	√	√		
g.s.m. inverse	√	√	√	√
Union		√		√
Intersection with regular set	√	√	√	√
Concatenation (·)				√
Kleene closure (*)				√
g.s.m. forward				√
Quotient with regular set (/)				√

IX. FUTURE PROBLEMS

There are various theorems about automata that have not been reflected in the results on balloon automata. For example, one-way deterministic pushdown automata are closed under complement. It is probably true that all common types of one-way or two-way deterministic automata are closed under complement, although proofs have not been published in all cases. Likewise, many one-way deterministic devices are closed under quotient with a regular set. Most one-way nondeterministic devices seem to be closed under reversal, and so on.

We therefore propose as an interesting and worthwhile problem, the question of putting additional restrictions on closed classes of balloon automata such that some or all of these results can be proven. Of course, the conditions must be liberal enough so that the usual automata are still modeled.

Second, it would be useful to have a model, like the balloon automaton, which could describe, as closed classes, such things as linear bounded automata and computational complexity classes. The properties of these classes deserve some treatment, and an approach similar to the one taken here might be a reasonable one.

It is hoped that the methods we have used to prove certain theorems plus the fact that we could not prove some others will shed some light on why some theorems are hard to prove, or visualize, while others are easy. Specifically, we have an indication as to why certain

theorems seem easier to prove for nondeterministic devices than deterministic.

REFERENCES

1. Turing, A. M., On Computable Numbers, With an Application to the *Entscheidungsproblem*, Proc. London Math. Soc., *42*, 1936, pp. 230–265.
2. Chomsky, N., Context Free Languages and Pushdown Storage, Quarterly Progress Report No. 65, Research Laboratory of Electronics, M.I.T., Cambridge, Massachusetts, 1962.
3. Evey, R. J., The Theory and Application of Pushdown Store Machines, Math. Linguistics and Automatic Translation, Report No. NSF-10, Computation Laboratory of Harvard Univ., May, 1963.
4. Gray, J., Harrison, M., and Ibarra, O., Two-Way Pushdown Automata, Univ. of California, Berkeley Dept. of Elect. Eng. Report, 1966.
5. Ginsburg, S. and Greibach, S., Deterministic Context Free Languages, Inform. Control, *9*, No. 6, December, 1966, pp. 620–648.
6. Schutzenberger, M., Finite Counting Automata, Inform. Control, *5*, No. 2, June, 1962, pp. 91–107.
7. Minsky, M., Recursive Unsolvability of Post's Problem of 'Tag' and Other Topics in the Theory of Turing Machines, Annals of Math., *74*, No. 3, November, 1961, pp. 437–455.
8. Ginsburg, S., Greibach, S., and Harrison, M., Stack Automata and Compiling, JACM, *14*, No. 1, January, 1967, pp. 172–201.
9. Ginsburg, S., Greibach, S., and Harrison, M., One-Way Stack Automata, JACM, *14*, No. 2, April, 1967, pp. 389–418.
10. Hopcroft, J. and Ullman J., Sets Accepted by One-Way Stack Automata are Context Sensitive. Submitted to Inform. Control.
11. Hopcroft, J. and Ullman J., Deterministic Stack Automata and the Quotient Operator, Submitted to JCSS. Also see Two Results on One-Way Stack Automata, 1967 IEEE Conf. Record Switching Auto. Theor., October, 1967.
12. Hopcroft, J. and Ullman J., Non-Erasing Stack Automata. JCSS, *1*, No. 2, August, 1967, pp. 166–186.
13. Aho, A., Indexed Grammars, — An Extension of Context Free Grammars, Princeton Univ. Ph.D. Thesis. Also 1967 IEEE Conf. Record Switching Auto. Theor., October, 1967.
14. Yamada, H., Real-Time Computation, and Functions not Real Time Computable, IRE Trans. Electron. Computer, *EC-11*, No. 6, December, 1962, pp. 753–760.
15. Hartmanis, J. and Stearns, R., On the Computational Complexity of Algorithms, Trans. Am. Math. Soc., *117*, No. 5, May, 1965.
16. Lewis, P., Stearns, R., and Hartmanis, J., Memory Bounds for the Recognition of Context Free and Context Sensitive Languages, 1965 IEEE Conf. Record Switching Circuit Theor. Logical Design, October, 1965, pp. 191–202.
17. Lewis, P., Stearns, R., and Hartmanis, J., Hierarchies of Memory Limited Computations, *Ibid.*, October 1965, pp. 179–189.
18. Hopcroft, J. and Ullman, J., Some Results on Tape Bounded Turing Machines. Submitted to JACM.
19. Ginsburg, S., Examples of Abstract Machines, IRE Trans. Electron. Computers, *EC-11*, 1962, pp. 132–135.
20. Rabin, M. and Scott, D., Finite Automata and Their Decision Problems, IBM J. Res. Devel., *3*, 1959, pp. 114–125.
21. Ginsburg, S. and Greibach, S., Abstract Families of Languages, 1967 IEEE Conf. Record Switching Auto. Theor., October, 1967.